A Practical Chosen Message Power Analysis Approach against Ciphers with the Key Whitening Layers

No Author Given

No Institute Given

Abstract. The key whitening is a technique intended to enhance the strength of a block cipher. Although some research work is involved the DPA against the key whitening layer in the compact architecture, there are no literatures dedicated in the influence of the key whitening layers in the loop architecture from the standpoint of DPA. In this paper, we propose a practical chosen message power analysis approach against the loop architecture of ciphers with the key whitening layers, thus the key whitening technique does not enhance the security of ciphers regard to DPA. Our approach follows a reduction strategy: we recover the whitening key in the general cipher with the key whitening layer and reduce other complicated key whitening layers to the general case. In order to further manifest the validity of the new approach, we carry extensive experiments on two ISO standardized ciphers CLEFIA and Camellia implemented in loop architecture on FPGA, and the keys are recovered as expected.

Keywords: DPA, key whitening, chosen message, loop architecture

1 Introduction

Key whitening is a technique intended to enhance the strength of a block cipher by adding key-relevant operations on plaintext and ciphertext without major changes to the algorithm [15, 16]. The key whitening layer consists of steps that combine the data with portions of the key before the first round and after the last round. The most common operation is XORing or modular adding the whitening key to the plaintext/ciphertext. The key whitening technique is adopted by many block ciphers, such as the ISO standardized Feistel-SP ciphers CLEFIA [2] and Camellia [3], and the lightweight ciphers DESL [15] and PRINCE [16].

Since first proposed by Kocher *et al.* in [1], DPA has proven to be a powerful method of side channel attack against many ciphers within these years. In general, DPA can only deal with a small fraction of the long secret key (*e.g.* several round key bits) through a divide-and-conquer strategy, and its validity is highly dependent on the specific implementation. These traditional cryptographic implementations (*i.e.*, compact architecture [18]) are easily compromised by the conventional DPA, where the hypothesis space of the secret key fraction is only

2^8 or less [17]. For instance, DES, AES and many other ciphers under the compact architecture have shown to be vulnerable to DPA [1, 4, 7, 9, 10]. The ciphers with the key whitening layers under same architecture are also easily compromised by DPA, because the key whitening layer is implemented independently of the substitution circuit [8].

With the advancement of circuit industry within recent years, a new architecture, *i.e.* loop architecture [18], is proposed, which computes a single round function in one clock cycle. Ciphers under the loop architecture are adopted in the higher computation speed and higher throughput application scenarios. Therefore, the capability of the loop implementation against the side channel attack has attracted researchers' great attention. It has been proved that the conventional DPA need much more power traces to analyze ciphers under loop architecture with very high computational complexity[13]. In order to deal with the loop architecture efficiently and practically, the adversary usually launches the chosen message DPA [12] instead.

However, it is quite a challenge to launch the DPA methodology against the loop hardware implementations of ciphers with the key whitening layers. The key whitening layer is usually implemented within the first or last round in the loop architecture, which can increase the difficulty of the DPA methodology. In this case, the power consumption of the whitening operation is hard to recognise from power traces, because the intermediate result of the whitening operation does not appear in registers or on bus as the case in [8]. Following the chosen message DPA [12], the adversary can only get the equivalent key (*i.e.* the value of "the round key add/xor the whitening key" as a unity) on the first/last round, but he would not be capable to directly determine either the whitening key or the round key. Unless the adversary is able to obtain the whitening key or the round key in some way, the adversary has to proceed DPA on several subsequent rounds to reveal enough round keys, which increases the difficulty and complexity of DPA. Therefore, the core issue in this case is how to reveal either the whitening key or the round key in the first/last round. To the best of our knowledge, no research about this issue has been reported in the literature. Consequently, launching the attack under the influence of the key whitening layers, is an interesting challenge.

In this paper, we propose a practical chosen message DPA approach to recover the whitening key through a reduction strategy. First, we recover the whitening key in the general cipher with the key whitening layer, by fully exploiting the relationship between the round key and the whitening key. Then we successfully reduce other complicated key whitening layers to the general case. As a result, we show that the key whitening technique does not enhance the security of ciphers from the standpoint of DPA.

We take the Feistel-SP ciphers with the key whitening layers as an instance, due to the most comprehensive cases of the key whitening layers in these ciphers (*i.e.*, the key whitening operation on the left branch, on the right branch, and on both branches). According to the relationship in the round function, our approach can launch chosen message DPA to efficiently reveal the whitening key on the left branch. When the whitening key is on the right branch, we are able to reduce the recovery of the whitening key on the right branch to the left branch case through an adaptive chosen message manner. When the whitening keys are on both branches, we can reduce the recovery of the whitening keys to the left branch case and the right branch case respectively. Furthermore, we perform extensive experiments on two ISO standardized ciphers CLEFIA and Camellia with loop FPGA implementations. Experimental results show that all bits of the keys in both ciphers can be recovered as expected.

The remainder of this paper is organized as follows. In Section 2, the preliminaries are briefly described. Section 3 illustrates the practical chosen message power analysis method on Feistel-SP ciphers with the key whitening layers. Section 4 elaborates the practical attacks on two loop FPGA implementations of CLEFIA-128 and Camellia-128 in order to prove the effectiveness of our approach. We discuss and summarize the paper in Section 5.

2 Preliminaries

2.1 The Compact and Loop Architecture

The hardware implementations of ciphers usually follow two architectures, compact architecture and loop architecture [18], in order to adapt to different application scenarios. In the embedded application scenario, the size, power consumption, and cost of the cryptographic device are tightly restrained. On the other hand, in the higher computation speed and higher throughput application scenario, the performance and efficiency of the cryptographic device are the most important indicators. The compact architecture usually takes several clock cycles to accomplish a single round computation of the cryptographic algorithm, such as reusing the single substitution circuit (*e.g.* Sbox) several times as a subloop instead of using their duplications concurrently. Therefore, the compact architecture, which sacrifices performance to less circuit components, is often applied to the embedded cryptographic device, such as smart cards and wireless sensor nodes.

On the other hand, the loop architecture defines the round function of the cryptographic algorithm as several consecutive operations, which means that a single round is computed in one clock cycle. Compared to the compact architecture, the loop architecture, which has higher throughput or less calculation time, is usually applied to the higher computation speed and higher throughput application scenario, such as the hardware security module in the cloud computing environment, the instant messenger system, the network authentication system and the network routing device. The loop architecture is implemented in ASIC or FPGA chips in the hardware security module, with the advancement of circuit industry within recent years.

2.2 DPA on Key Whitening Layer

The key whitening layers can be compromised by DPA, as long as the adversary is able to locate the power consumption of the whitening operation on power traces. In the compact scenario, the power consumption of the whitening operation is convenient to be observed due to the independent implementation of the key whitening layer. Consequently, the adversary can reveal the whitening key through direct analysis of the power consumption of the whitening operation [8, 11]. Unfortunately, the above strategy seems hard to apply in the loop scenario, because the whitening operations are implemented within the first/last round. It leads to much lower signal-to-noise of the slight power consumption which is too difficult to be detected. In this case, the adversary can only get the equivalent key (*i.e.* the value of "the round key adding the whitening key" as a unity) in DPA, but he would not be capable to directly determine either the whitening key or the round key. Therefore, how to reveal the whitening key or the round key in the loop architecture remains an open problem.

2.3 Feistel-SP Structure

The Feistel network is one of the most famous architectures in symmetric cryptography. According to the classifications of [5], the Feistel network has several derivatives, namely, the unbalanced Feistel network, the alternating Feistel network, the numeric Feistel network, and the famous type-1, type-2, and type-3 Feistel networks. Many practical block ciphers utilize the Feistel networks including DES (plain), Skipjack (unbalanced), BEAR/LION (alternating), CAST (type-1), CLEFIA(type-2) and MARS(type-3).

A typical SP type function often consists of three operations, *i.e.*, subkey addition, substitution, and permutation. In the subkey addition, a subkey is XORed to the state. The substitution is applied by Sbox-like non-linear bijection. In the permutation, a linear bijection (generally an MDS multiplication) is performed. Let $S_1, S_2, \cdots, S_t : \{0,1\}^s \longrightarrow \{0,1\}^s$ be non-linear bijections, $P : \{0,1\}^{st} \longrightarrow \{0,1\}^{st}$ be a linear bijection, $k = (k_1, k_2, \cdots, k_t)$ is the round key, then the round function $F : \{0,1\}^{st} \times \{0,1\}^{st} \longrightarrow \{0,1\}^{st}$ of SP type is defined by $F(x,k) = P(S_1(x_1 \oplus k_1), S_2(x_2 \oplus k_2), \cdots, S_t(x_t \oplus k_t))$. The notations s and t represent the size of the non-linear bijection and the number of the non-linear bijections, respectively.

In the Feistel network, the core of the underlying round function is referred to as the F-function. In order to combine the advantages of SPN structures, the Feistel-SP ciphers use well-designed SPN functions as the F-functions. The typical round function of the Feistel-SP structure is shown in Fig.1.

3 The Design of Our Approach

In this section, we describe the details of our practical chosen message DPA method on the loop architecture of Feistel-SP ciphers with the key whitening layers. Firstly, sharing the similar idea of chosen message DPA [12], we put forward the chosen message DPA which is also suitable for the case of Feistel-SP structure without the key whitening layer. Secondly, we analyze the difficulty to reveal the whitening key in the loop architecture directly. Then we propose

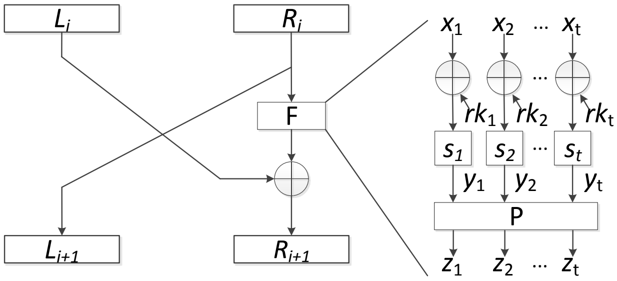


Fig. 1. The typical round function of the Feistel-SP structure

an efficient approach to recover the whitening key through a reduction strategy. More precisely, we show how to recover the whitening key on the left branch, and reduce the recovery of the whitening key on the right branch to the left branch case through an adaptive chosen message manner. Combining these two strategies, we manage to perform practical DPA on loop architecture of Feistel-SP structure with the key whitening layer.

3.1 The Chosen Message DPA on the Loop Architecture of Feistel-SP

The typical loop implementation of Feistel-SP is shown in Fig.1. Let L_i and R_i denote the left and right branches of the i-th round input respectively, and the size of the Sbox is 8-bit. The right branch R_i can be further split into t 8-bit cells, namely, $R_i = x_1||x_2||\cdots||x_t$. Each 8-bit cell x_j is first XORed with the corresponding 8-bit round key rk_j, then all t cells are processed with t parallel Sboxes $S_1, S_2, \cdots, S_t$ [1]. Let $y_1||y_2||\cdots||y_t$ denote the output of the Sbox layer, the linear permutation P (normally multiplication with an MDS matrix) updates the state $y_1||y_2||\cdots||y_t$, and $z_1||z_2||\cdots||z_t$ is the output. The right branch of the i-th round output R_{i+1} is then calculated by XORing L_i and the output of P, while the left branch L_{i+1} is updated by directly assigning the value of R_i. The above procedures can be described as

$$\begin{aligned} L_{i+1} &= R_i, \\ R_{i+1} &= F(R_i, rk) \oplus L_i \\ &= P(S_1(x_1 \oplus rk_1), S_2(x_2 \oplus rk_2), \cdots, S_t(x_t \oplus rk_t)) \oplus L_i. \end{aligned} \tag{1}$$

In the scenario of loop architecture, Feistel-SP treats the round function as several consecutive operations, thus no intermediate result except R_{i+1} is written into registers in the loop implementations. The power consumption of y_j is much less than R_{i+1} and hard to be observed. Therefore, we are only able to attack at this point when the data R_{i+1} is being written into registers as shown in Fig.2.

[1] The Sboxes can be identical or distinct.

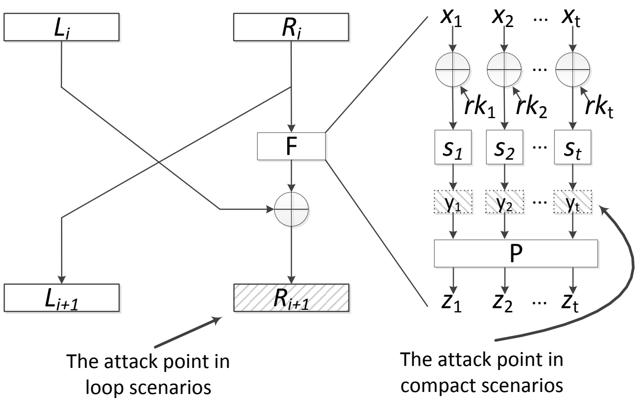


Fig. 2. Different DPA attack points of Feistel-SP

As shown in Fig.2, regarding a specific cell y_j, the linear permutation P can be represented as follows:

$$\begin{aligned} P(y_1, y_2, \cdots, y_{j-1}, y_j, y_{j+1}, \cdots, y_t) = &\ P(0, 0, \cdots, 0, y_j, 0, \cdots, 0)\oplus \\ &\ P(y_1, y_2, \cdots, y_{j-1}, 0, y_{j+1}, \cdots, y_t) \\ = &\ WP^j \oplus CP^j, \end{aligned}$$

where the superscript j indicates the function focusing on the cell y_j, and WP^j and CP^j denote the two components of the right half of the equation respectively. The above equation can be rewritten in byte-wise form as follows:

$$P_{[1]}||P_{[2]}||\cdots||P_{[t]} = (WP^j_{[1]}||WP^j_{[2]}||\cdots||WP^j_{[t]}) \oplus (CP^j_{[1]}||CP^j_{[2]}||\cdots||CP^j_{[t]})$$

where the subscript $[k]$ indicates the k-th byte output, and $WP^j_{[k]}$ and $CP^j_{[k]}$ denote the k-th bytes of WP^j and CP^j respectively. If we fix the values of all y_k where $k \neq j$, and keep y_j as a variable, then $WP^j_{[j]}$ can be seen as a function of y_j and $CP^j_{[j]}$ is a constant. For convenience, we use $P'_{[j]}$ and $mask_{[j]}$ to represent $WP^j_{[j]}$ and $CP^j_{[j]}$ respectively.

Thus, the adversary chooses the specific byte of plaintext message, which corresponds the j-th byte of the target intermediate variable, while fixing other bytes of the plaintext message. The target byte is dependant on two unknown one-byte constant parameters, *i.e.*, the subkey rk_j and the $mask_{[j]}$ generated by P. Therefore, the size of guessed parameters from the whole round key is decreased to a pair of 8-bit values, *i.e.*, the hypothesis space of the secret value falls to 2^{16}, and the input space of the plaintext message is decreased to 2^8, which is suitable for practical DPA. Consequently, by alternately choosing the corresponding plaintext message byte for all possible positions, we can use the DPA attacking model shown in Fig.3 to launch DPA and recover all t bytes of the round key (for the sake of simplicity, we assume the size of the round key is 32 bits).

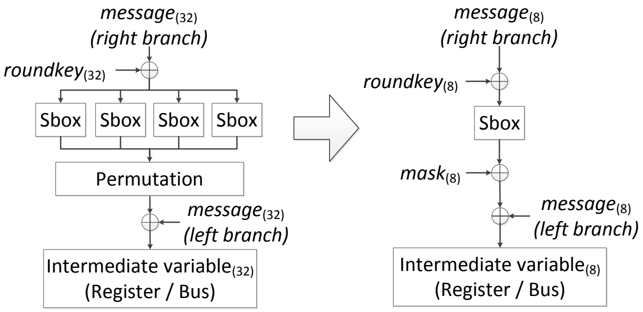


Fig. 3. DPA model of Feistel-SP in loop scenario

3.2 The Difficulty to Reveal The Whitening Key in the Loop Scenario

The whitening keys are generally used before the first round and after the last round. After the key whitening operations, the inputs (outputs) to the first (last) round are covered by the unknown whitening key from the plaintexts (ciphertexts). It seems that such operations would increase the size of unknown parameters, and raise the difficulty to launch a DPA attack. However, since the encryption and decryption of Feistel-SP ciphers follow similar procedures and both the pre-whitening and the post-whitening keys are almost equivalent from the perspective of DPA, we only discuss the encryption procedure in the first round.

Let ML (resp. MR) denote the left (resp. right) message branch, and wkL (resp. wkR) denote the left (resp. right) whitening key. As shown in Fig.4, the whitening keys can be applied on the left branch, the right branch or both branches, which correspond to three types of whitening operations.

There are two main difficulties to apply DPA to reveal the whitening key in loop scenario. The first difficulty is that the power consumption of the whitening operation is hard to detect from power traces similar as Section 3.1. In the loop hardware implementation, the whitening operation and the round key addition operation are usually combined as one operation (*i.e.* $Message \oplus wk \oplus rk$), which is implemented by 3-input XOR gate or LUT. More precisely, there is no standalone whitening operation in the Feistel-SP computing procedure, thus the power consumption of the whitening operation is hard to detect. Therefore, the existing method against the whitening key as mentioned in [8] is not suitable.

Moreover, although we could choose R_{i+1} as the attack point, the whitening key is difficult to be separated from the round key and other intermediate

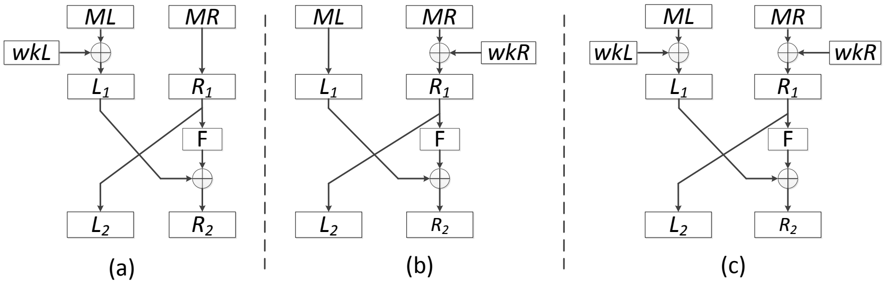


Fig. 4. The whitening key on (a) left branch (b) right branch (c) both branches

variables by DPA. We assume that there are whitening keys on both branches. Due to the effect of whitening keys wkL and wkR, the DPA attacking model of Feistel-SP with whitening key in loop scenario is shown in Fig.5(a). For the loop scenario, we use $rk \oplus wkR$ as the equivalent key and $mask \oplus wkL$ as the equivalent mask, thus the model is changed from Fig.5(a) to Fig.5(b). However, although we can recover the equivalent key and the equivalent mask by DPA, we are unable to directly determine the values of the whitening keys from the equivalent key and the equivalent mask.

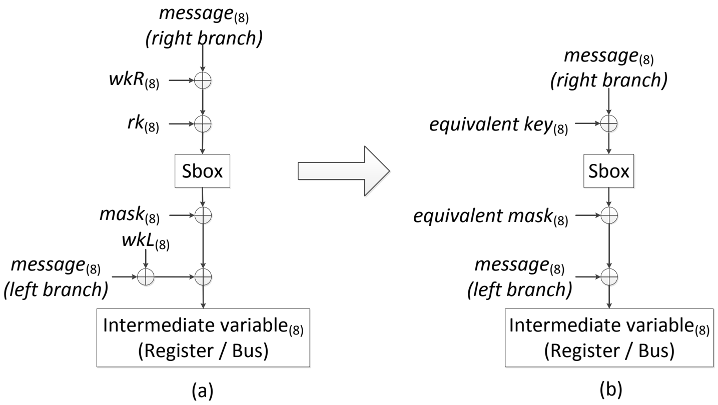


Fig. 5. Chosen Message DPA model of Feistel-SP whitening key in the loop scenario

3.3 Recovery of wkL

When we choose R_{i+1} as the attack point, the main drawback for DPA is the difficulty to separate the whitening key from the equivalent key and the equivalent mask. Luckily, we can achieve this goal through a reduction strategy. More precisely, we can efficiently recover the whitening key on the left branch wkL, and reduce the recovery of the whitening key on the right branch to the left branch case.

The recovery of wkL is based on the full exploitation of the complex relationship between the equivalent key and the equivalent mask with a chosen message DPA method. Hereafter we use subscript $[i]$ $(1 \leq i \leq t)$ to indicate the i-th byte of the corresponding variable or the output of one function, and use notation rk_j $(j \geq 1)$ to represent the round key in the j-th round of Feistel-SP. According to Fig.4(a), two branches of the first round input $L_1||R_1$ can be described by:

$$L_1||R_1 = (ML \oplus wkL)||MR. \tag{2}$$

Now, we will focus on the attack point R_2 as shown in Fig.2. Equation.1 can be rewritten as:

$$\begin{aligned} R_2 = \; & F(R_1, rk_1) \oplus L_1 \\ = \; & P(S_1(MR_{[1]} \oplus rk_{1,[1]}), S_2(MR_{[2]} \oplus rk_{1,[2]}), \\ & \cdots, S_t(MR_{[t]} \oplus rk_{1,[t]})) \oplus wkL \oplus ML. \end{aligned} \tag{3}$$

where $MR = MR_{[1]}||MR_{[2]}||\cdots||MR_{[t]}$, and $rk_1 = rk_{1,[1]}$ $||rk_{1,[2]}||\cdots||rk_{1,[t]}$.

We focus on the first byte of R_2, and fix all $MR_{[j]}$ $(2 \leq j \leq t)$ to constants, that leads to the constant output of S_j. Thus, Equation.3 can be rewritten in byte-wise form:

$$\begin{aligned} R_{2,[1]} = \; & F_{[1]}(R_1, rk_1) \oplus L_{1,[1]} \\ = \; & P_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]}), S_2(MR_{[2]} \oplus rk_{1,[2]}), \\ & \cdots, S_t(MR_{[t]} \oplus rk_{1,[t]})) \oplus wkL_{[1]} \oplus ML_{[1]} \\ = \; & P'_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]})) \oplus mask_{1,[1]} \oplus wkL_{[1]} \oplus ML_{[1]} \\ = \; & P'_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]})) \oplus MASK_{1,[1]} \oplus ML_{[1]}, \end{aligned} \tag{4}$$

with $ML = ML_{[1]}||ML_{[2]}||\cdots||ML_{[t]}$, $wkL = wkL_{[1]}||wkL_{[2]}||\cdots||wkL_{[t]}$. Moreover, $mask_1 = mask_{1,[1]}||mask_{1,[2]}||\cdots||mask_{1,[t]}$ is the intermediate variable which is generated in the first round. According to Section 3.1, $mask_{1,[j]}$ is a byte constant value if all MR_k $(k \neq j)$ are fixed since the round key rk_1 is pre-assigned, and the equivalent mask $MASK_1 = mask_1 \oplus wkL$.

At this time, $R_{2,[1]}$ is highly related to $rk_{1,[1]}$, and $R_{2,[2]}, R_{2,[3]}, \cdots, R_{2,[t]}$ will be treated as noise. Now, we can launch DPA against $R_{2,[1]}$ by enumerating 8-bit $MR_{[1]}$ while fixing other bits of MR and ML. Thus, both 8-bit $rk_{1,[1]}$ and 8-bit $MASK_{1,[1]}$ are revealed by DPA, where the possible hypotheses space is 2^{16} and the possible input space of random test vector $MR_{[1]}$ is only 2^8. With the same approach, we could analyze $R_{2,[2]}, R_{2,[3]}, \cdots, R_{2,[t]}$ byte by byte, and reveal the values of $rk_{1,[2]}, rk_{1,[3]}, \cdots, rk_{1,[t]}$ and $MASK_{1,[2]}, MASK_{1,[3]}, \cdots, MASK_{1,[t]}$.

According to the complex relationship between rk_1 and $mask_{1,[j]}$, we could iteratively calculate each of $mask_{1,[j]}$ by all bytes of rk_1. More precisely, according to Equation.4, $mask_{1,[1]}$ is calculated by:

$$\begin{aligned} mask_{1,[1]} = \; & P'_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]})) \oplus P_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]}), \\ & S_2(MR_{[2]} \oplus rk_{1,[2]}), \cdots, S_t(MR_{[t]} \oplus rk_{1,[t]})). \end{aligned} \tag{5}$$

$mask_{1,[j]}$ $(j \in [2, t])$ is calculated similarly. Then the values of all $wkL_{[k]}$ $(k \in [1, t])$ is iteratively calculated by $wkL_{[k]} = MASK_{1,[k]} \oplus mask_{1,[k]}$. Thus, the whitening key wkL is recovered, and our method is suitable for launching DPA to recover the whitening key on the left branch of Feistel-SP in loop architecture.

3.4 Recovery of *wkR*

According to Section 3.3, we reveal the round key rk_1 and the equivalent mask $MASK_1$ in the first round, and then successfully derive the left branch whitening key wkL from above two parameters. However, due to Fig.4(b) and Fig.5, the right branch whitening key wkR is hard to be distinguished from the equivalent key, because both wkR and rk_1 have exactly the same effects on the SP type F-function in the first round of Feistel-SP ciphers. More precisely, according to Equation.4, the first byte of R_2 in this case would be rewritten as:

$$\begin{aligned} R_{2,[1]} = \; & F_{[1]}(R_1, rk_1) \oplus L_{1,[1]} \\ = \; & P_{[1]}(S_1(MR_{[1]} \oplus wkR_{[1]} \oplus rk_{1,[1]}), S_2(MR_{[2]} \oplus wkR_{[2]} \oplus rk_{1,[2]}), \\ & \cdots, S_t(MR_{[t]} \oplus wkR_{[t]} \oplus rk_{1,[t]})) \oplus ML_{[1]} \\ = \; & P'_{[1]}(S_1(MR_{[1]} \oplus wkR_{[1]} \oplus rk_{1,[1]})) \oplus mask_{1,[1]} \oplus ML_{[1]} \\ = \; & P'_{[1]}(S_1(MR_{[1]} \oplus ek_{1,[1]})) \oplus mask_{1,[1]} \oplus ML_{[1]}. \end{aligned} \tag{6}$$

with $wkR = wkR_{[1]}||wkR_{[2]}|| \cdots ||wkR_{[t]}$, and the equivalent key $ek_1 = rk_1 \oplus wkR$, and $mask_{1,[1]}$ is a byte constant value if $MR_{[2]}, MR_{[3]}, \cdots, MR_{[t]}$ are fixed since the equivalent key ek_1 is pre-assigned. In this scenario, we can only reveal each byte of ek_1 and $mask_1$ with the approach proposed in Section 3.3, and we are unable to split wkR from ek_1.

Therefore, we have to find relations between the case of the whitening key on the right branch and the case of the whitening key on the left branch in order to recover wkR. We focus on the second round of Feistel-SP. As shown in Fig.6(a), the whitening key wkR is mixed up with $mask_2$ in the second round of Feistel-SP, and we choose R_3 as the attack point in the second round. Equation.3 and Equation.4 can be rewritten as:

$$\begin{aligned} R_3 = \; & F(R_2, rk_2) \oplus L_2 \\ = \; & P(S_1(R_{2,[1]} \oplus rk_{2,[1]}), S_2(R_{2,[2]} \oplus rk_{2,[2]}), \\ & \cdots, S_t(R_{2,[t]} \oplus rk_{2,[t]})) \oplus wkR \oplus MR. \end{aligned} \tag{7}$$

$$\begin{aligned} R_{3,[1]} &= F_{[1]}(R_2, rk_2) \oplus L_{2,[1]} \\ &= P_{[1]}(S_1(R_{2,[1]} \oplus rk_{2,[1]}), S_2(R_{2,[2]} \oplus rk_{2,[2]}), \\ &\quad \cdots, S_t(R_{2,[t]} \oplus rk_{2,[t]})) \oplus wkR_{[1]} \oplus MR_{[1]} \\ &= P'_{[1]}(S_1(R_{2,[1]} \oplus rk_{2,[1]})) \oplus mask_{2,[1]} \oplus wkR_{[1]} \oplus MR_{[1]} \\ &= P'_{[1]}(S_1(R_{2,[1]} \oplus rk_{2,[1]})) \oplus MASK_{2,[1]} \oplus MR_{[1]}. \end{aligned} \tag{8}$$

where $mask_{2,[1]}$ is a byte constant value if $R_{2,[2]}, R_{2,[3]}, \cdots, R_{2,[t]}$ are fixed since the round key rk_2 is pre-assigned. Therefore, as shown in Fig.6(b), if we can control the input of the second round $L_2||R_2$ as the plaintext message $ML||MR$, we will reduce the case of the whitening key on the right branch in the second round to the case of the whitening key on the left branch in the first round.

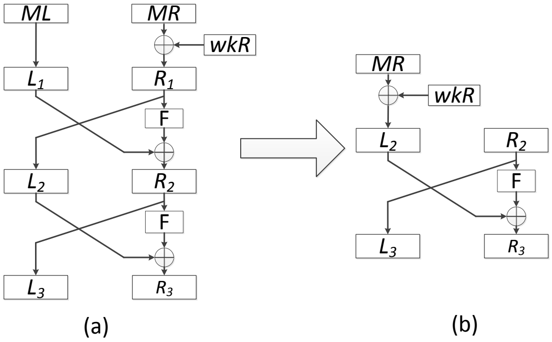


Fig. 6. The reduce from the right branch case to the left branch case

Fortunately, we can control the input of the second round $L_2||R_2$ in an adaptive chosen message manner. Firstly, we reveal ek_1 in the first round. Then based on the values of $L_2||R_2$ which meets the chosen message requirements, we calculate the corresponding plaintext message with ek_1 through the inverse transformation of the Feistel-SP round function. Finally, we could establish the chosen plaintext set which is suitable for the second round and reduce the case of wkR to the case of wkL, and reveal wkR for the second round of Feistel-SP.

On Recovering the Whitening Keys on Both Branches. Our approach is also suitable for Feistel-SP ciphers with the key whitening layer on both branches, as shown in Fig.4(c). The specific procedures of this scenario are as follows:

- **Step 1. Reveal the Whitening Key wkL in the First Round.** The first step of our method is to reveal wkL in the first round. We establish the chosen plaintext set which enumerates all possible values of the target bytes ($MR_{[1]}$), and fixes other bytes to constants. Then, we repeat the DPA attack against R_2 several times to recover all bytes of ek_1 and $MASK_1$.

Finally, we calculate $mask_1$ by ek_1 and reveal wkL with the equation $wkL = MASK_1 \oplus mask_1$.

- **Step 2. Reveal the Whitening Key wkR in the Second Round.** The second step of our method is to establish the chosen plaintext set which is used for the second round, through an adaptive chosen message manner. According to Section 3.4, it can be done with similar strategy of Step 1 to recover wkR.

4 Applications

We apply these techniques to two typical Feistel-SP ciphers, CLEFIA-128 [2] and Camellia-128 [3], to verify the effectiveness of our method. We implement both ciphers with the loop architecture on a Virtex-5 Xilinx FPGA on SASEBO-GII board. Pearson Correlation Coefficient based Power Analysis (CPA) is applied during the security analysis. The aim is to recover the master keys in CLEFIA-128 and Camellia-128, and the master keys are recovered as expected in both experiments, thus manifesting the correctness of our approach.

We illustrate the applications of our methods to CLEFIA-128 in the following part. Due to the space limitation, details of the attack on Camellia-128 are provided in the appendix.

4.1 Specification to CLEFIA-128

CLEFIA-128 is a type-2 Feistel-SP cipher proposed at FSE 2007 by Shirai *et al.*. It is standardized by ISO [19] as a lightweight cipher. It encrypts a 128-bit plaintext into a 128-bit ciphertext with a 128-bit master key after applying the round function 18 times, as shown in Fig.7.

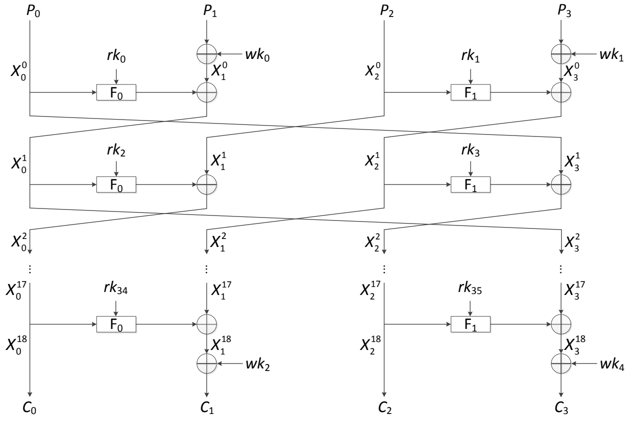


Fig. 7. CLEFIA encryption algorithm for 128-bit key

Let P and K be the 128-bit plaintext and the master key respectively. Thirty-six 32-bit round keys $rk_0, rk_1, \cdots, rk_{35}$ and four 32-bit whitening keys wk_0, wk_1, wk_2, wk_3 are generated from K. These whitening keys are defined as $wk_0||wk_1||wk_2||wk_3 = K$ according to the key schedule.

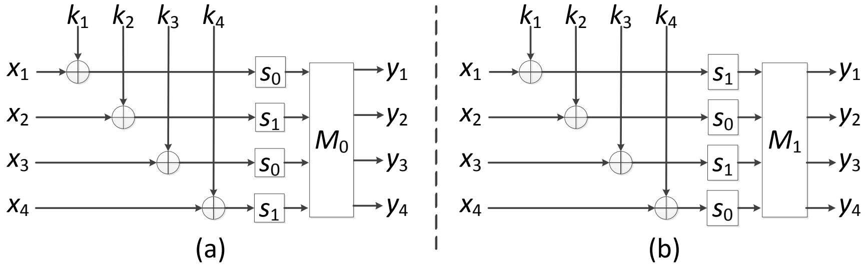


Fig. 8. CLEFIA SP type F-functions for (a)F_0 (b)F_1

Let $X_0^i||X_1^i||X_2^i||X_3^i (0 \leq i \leq 17)$ be an internal input state in each round. The plaintext is loaded into $P_0||P_1||P_2||P_3$. Next, X_1^0 and X_3^0 are updated by the pre-whitening layer, that is $(X_0^0, X_1^0, X_2^0, X_3^0) = (P_0, P_1 \oplus wk_0, P_2, P_3 \oplus wk_1)$. Then, the internal state is updated by the following computation up to the second last round (for $1 \leq i \leq 17$);

$$X_0^i = X_1^{i-1} \oplus F_0(X_0^{i-1}, rk_{2i-2}), X_1^i = X_2^{i-1},$$
$$X_2^i = X_3^{i-1} \oplus F_1(X_2^{i-1}, rk_{2i-1}), X_3^i = X_0^{i-1}.$$

Two SP type F-functions F_0 and F_1 consist of a 32-bit round key addition, an S-box transformation, and a multiplication by an MDS matrix, as shown in Fig.8. Four parallel 8-bit Sboxes are applied, followed with an MDS multiplication in each of SP type F-functions. In addition, the MDS matrices for two SP type F-functions are different.

In the last round, $X_0^{18}||X_1^{18}||X_2^{18}||X_3^{18}$ is computed by

$$X_0^{18} = X_0^{17}, X_1^{18} = X_1^{17} \oplus F_0(X_0^{17}, rk_{34}),$$
$$X_2^{18} = X_2^{17}, X_3^{18} = X_3^{17} \oplus F_1(X_2^{17}, rk_{35}).$$

Finally, C_1 and C_3 are updated by the post-whitening layer, *i.e.*, $(C_0, C_1, C_2, C_3) = (X_0^{18}, X_1^{18} \oplus wk_2, X_2^{18}, X_3^{18} \oplus wk_3)$, and $C_0||C_1||C_2||C_3$ is the final ciphertext.

4.2 Attack of CLEFIA-128

CLEFIA adopts 4-branch Type-2 Feistel network but uses two different diffusion matrices for the diffusion switching mechanism. It has the key whitening layer and only the left branch of the plaintext blocks is XORed with the whitening key. Our aim is to reveal the master keys through the recovery of the whitening keys.

Hereafter we use the notations P_j and C_j for CLEFIA-128 to represent the plaintext and ciphertext in the j^{th} branch, X_j^i to represent the state in the j^{th} branch immediately after the round operation in round i, X_j^0 to represent the output of the key whitening layer before the first round. We use the notations F_0, F_1, S_0, S_1, M_0, and M_1 to further distinguish the different round functions of CLEFIA-128. The subscript $[n]$ indicates the n^{th} byte of the state.

To reveal the master key K of CLEFIA-128, we focus on the whitening key. As shown in Fig.7, the four 32-bit whitening keys wk_0, wk_1, wk_2, wk_3 are generated from K. These whitening keys are defined as $wk_0||wk_1||wk_2||wk_3 = K$. Thus, we do not need to calculate the inverse transformation of CLEFIA key schedule to reveal K, if we get the whitening key value. There are two key whitening layers in CLEFIA-128, the pre-whitening before the first round and the post-whitening after the last round. According to the Section 3.3, CLEFIA-128 belongs to the case of the whitening key on the left branch. Hence, our attack target is the first round and the last round of CLEFIA-128.

In order to recover both the pre-whitening and the post-whitening keys, the attack should be conducted in both the encryption and the decryption directions respectively. However, since the encryption and decryption of CLEFIA follow similar procedures and F_0 and F_1 are almost equivalent from the perspective of DPA, the attack procedures for recovering the whitening keys are almost identical. Thus we only describe the detailed process to recover wk_0.

The whitening key wk_0 is only XORed with 32-bit plaintext block P_1, and rk_0 is the round key. According to the specification of CLEFIA-128 and Equation.1, the 32-bit output X_0^1 is described by

$$\begin{aligned} X_0^1 = & \ F_0(X_0^0, rk_0) \oplus X_1^0 \\ = & \ M_0(S_0(P_{0,[1]} \oplus rk_{0,[1]}), S_1(P_{0,[2]} \oplus rk_{0,[2]}), \\ & \ \cdots, S_1(P_{0,[4]} \oplus rk_{0,[4]})) \oplus wk_0 \oplus P_1. \end{aligned} \tag{9}$$

Focusing on the first byte of X_0^1, Equation.9 could be rewritten as

$$\begin{aligned} X_{0,[1]}^1 &= S_0(P_{0,[1]} \oplus rk_{0,[1]}) \oplus mask_{0,[1]} \oplus wk_{0,[1]} \oplus P_{1,[1]} \\ &= S_0(P_{0,[1]} \oplus rk_{0,[1]}) \oplus MASK_{0,[1]} \oplus P_{1,[1]}. \end{aligned} \tag{10}$$

where $mask_{0,[1]}$ is generated by $S_1(P_{0,[2]} \oplus rk_{0,[2]})$, $S_0(P_{0,[3]} \oplus rk_{0,[3]})$, and $S_1(P_{0,[4]} \oplus rk_{0,[4]})$, and $MASK_0$ is defined as $MASK_0 = mask_0 \oplus wk_0$.

Therefore, we can reveal the values of rk_0 and $MASK_0$ by chosen message DPA method as mentioned in Section 3.3. Then, we calculate $mask_0$ by rk_0 and reveal wk_0 with the equation $wk_0 = MASK_0 \oplus mask_0$. Thus, wk_1, wk_2, and wk_3 can be revealed by similar procedures. After deriving all the whitening keys, the master key K can be easily derived since $K = wk_0||wk_1||wk_2||wk_3$.

We preset the master key K as four consecutive 32-bit words denoted in hex form, FFEEDDCC-BBAA9988-77665544-33221100, in our FPGA implementation with the loop architecture. Thus, it is obvious that $wk_0 =$ FFEEDDCC and $rk_0 =$

F3E6CEF9. We use the attack result of the first Sbox (*i.e.*, $rk_{0,[1]}$ and $MASK_{1,[1]}$) as an example. In Fig.9(a), there are two max points F3D4 and F32B whose correlation coefficients are 0.0743 and -0.0743 respectively. According to the knowledge of the FPGA platform, the power consumption of the FPGA platform has a negative correlation with the Hamming distance power model. Thus, F32B is the attack result (*i.e.*, $rk_{0,[1]} =$ F3 and $MASK_{1,[1]} =$ 2B), which is revealed within 10000 power traces as shown in Fig.9(b).

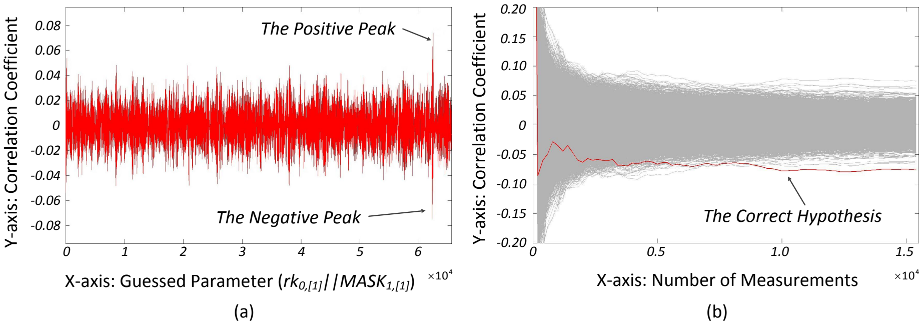


Fig. 9. Correlation traces in different hypothesis and different number of measurements on CLEFIA

We repeat the above procedure 3 times more, and all bytes of rk_0 and $MASK_1$ are revealed as $rk_0 =$ F3E6CEF9 and $MASK_1 =$ 2BF18258. Thus, the value of $mask_1$ is D41F5F94, which is calculated by the round key rk_0. And the wk_0 is FFEEDDCC, calculated by $mask_1 \oplus MASK_1$. The remaining parts of whitening keys are calculated by similar way. Now, we calculate K is FFEEDDCC-BBAA9988-77665544-33221100, which is identical with the preset master key.

5 Conclusion and Discussion

In this paper, we propose a practical chosen message DPA approach to recover the whitening key through a reduction strategy, and show that the key whitening technique does not enhance the security of ciphers from the standpoint of DPA. Then we apply our method to CLEFIA-128 and Camellia-128, and the master keys are recovered as expected. Following the results presented in this work, several problems which are worth further investigations:

- **Optimizations.** The first natural question emerges is whether our method can be further optimized. One possible direction of improvement is taking advantage of more powerful DPA method, such as the adaptive strategy in [6, 14] and the multiple CPA in [12], to discriminate the correct hypothesis from the key candidates in a more efficient way. Other potential optimizations are also possible directions for future research.

- **Countermeasures.** Next to optimality, another important question is to determine the countermeasures against such attack. Our method is suitable for the unprotected loop hardware implementations. Several common countermeasures on the compact architecture [17], can be considered to apply to the loop architecture in order to resist our approach, while the resource consumption will have a corresponding increase. Moreover, the countermeasures based on the mask methodology should be used with caution on the loop implementations of ciphers, because the mask countermeasures usually lead to slow computation speed. Considering the limitation of high computation speed and high throughput in the application scenario, the loop implementations of ciphers must keep the high performance, when the countermeasures against such attack are applied on these implementations. Thus, a trade-off between performance and security should be considered by the vendor.

References

1. Kocher,P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) Annual Cryptology Conference, CRYPTO 1999, Springer, Heidelberg, LNCS, vol. 1666, pp. 388-397 (1999).
2. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) International Workshop on Fast Software Encryption 2007, Springer, Heidelberg, LNCS, vol. 4593, pp. 181-195 (2007).
3. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-bit block cipher suitable for multiple platforms. In: Stinson, D.R., Tavares, S. (eds.) International Workshop on Selected Areas in Cryptography 2000, Springer, Heidelberg, LNCS, vol. 2012, pp. 41-54 (2001).
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) International Workshop on Cryptographic Hardware and Embedded Systems 2004, Springer, Heidelberg, LNCS, vol. 3156, pp. 16-29 (2004).
5. Hoang, V.T., Rogaway, P.: On Generalized Feistel Networks. In: Rabin, T. (ed.) Annual Cryptology Conference, CRYPTO 2010, Springer, Heidelberg, LNCS, vol. 6223, pp. 613-630 (2010).
6. Kopf, B., Basin, D.A.: An information-theoretic model for adaptive side-channel attacks. In: Ning, P., De Capitani Vimercati di, S., Syverson, P.F. (eds.) Proceedings of the 14th ACM conference on Computer and communications security, ACM-CCS 2007. ACM, pp. 286-296 (2007).
7. Kim, Y., Ahn, J., Choi, H.: Power and Electromagnetic Analysis Attack on a Smart Card Implementation of CLEFIA. In: International Conference on Security and Management, SAM 2013 (2013).
8. Akkar, M.-L., Bevan, R., Dischamp, P., Moyart, D.: Power Analysis, What Is Now Possible.... In: Okamoto, T. (ed.) International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2000, Springer, Heidelberg, LNCS, vol. 1976, pp. 489-502 (2000).
9. Lu, Y., O'Neill, M.P., McCanny, J.V.: Differential Power Analysis resistance of Camellia and countermeasure strategy on FPGAs. In: International Conference on Field-Programmable Technology 2009, pp: 183-189 (2009).

10. Xiao, L., Heys, H.: A Simple Power Analysis Attack against the Key Schedule of the Camellia Block Cipher. In: Information Processing Letters, Elsevier, vol. 95, pp 409-412 (2005).
11. Bayrak, A.G., Regazzoni, F., Novo, D., Ienne, P.: Sleuth: automated verification of software power analysis countermeasures. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 293-310. Springer, Heidelberg (2013)
12. Moradi, A., Schneider, T.: Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series. In: Standaert, F.-X., Oswald, E. (eds.) Constructive Side-Channel Analysis and Secure Design 2016, Springer, Heidelberg, LNCS, vol. 9689, pp. 71-87 (2016).
13. Moradi, A., Kasper, M., Paar, C.: Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures - An Analysis of the Xilinx Virtex-4 and Virtex-5 Bitstream Encryption Mechanism. In: Dunkelman, O. (ed.) Cryptographers' Track at the RSA Conferenc 2012, Springer, Heidelberg, LNCS, vol. 7178, pp. 1-18 (2012).
14. Veyrat-Charvillon, N., Standaert, F.-X.: Adaptive Chosen-Message Side-Channel Attacks. In: Zhou, J., Yung, M. (eds.) International Conference on Applied Cryptography and Network Security 2010, Springer, Heidelberg, LNCS, vol. 6123, pp. 186-199 (2010).
15. Poschmann, A., Leander, G., Schramm, K., Paar, C.: New Light-Weight DES Variants Suited for RFID Applications. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196-210. Springer, Heidelberg (2007).
16. Borghoff, J., et al.: PRINCE - A low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208-225. Springer, Heidelberg (2012).
17. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks: revealing the secrets of smart cards. Springer, New York, ISBN: 978-0-387-30857-9 (2007).
18. Rodríguez-Henríquez, F., Saqib, N.A., Díaz-Pèrez, A., Koc, Ç.K.: Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology). Springer-Verlag New York, Inc., Secaucus (2006).
19. ISO/IEC 29192-2:2011. Information technology - Security techniques - Lightweight cryptography-Part 2: Block ciphers (2011).
20. ISO/IEC 18033-3:2010. Information technology - Security techniques - Encryption Algorithms-Part 3: Block ciphers (2010).
21. New European Schemes for Signatures, Integrity, and Encryption(NESSIE). NESSIE PROJECT ANNOUNCES FINAL SELECTION OF CRYPTO ALGORITHMS (2003).
22. Cryptography Research and Evaluation Committees (CRYPTREC). e-Government recommended ciphers list (2003).

A Application to Camellia-128

A.1 Specification of Camellia-128

Camellia-128, proposed at SAC 2000 by Aoki *et al.* [3], was jointly designed by NTT and Mitsubishi Electric Corporation. It is widely acknowledged and recommended by ISO [20], NESSIE [21], and CRYPTREC [22]. It encrypts a 128-bit plaintext into a 128-bit ciphertext with a 128-bit master key after applying the round function 18 times, as shown in Fig.10.

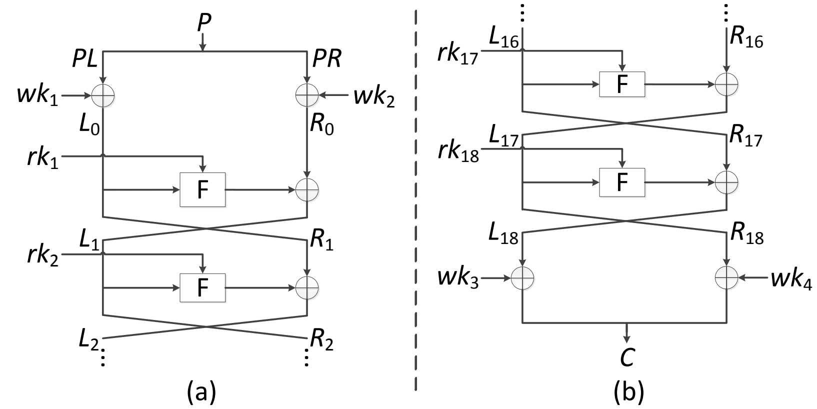


Fig. 10. Camellia encryption algorithm for (a)First two rounds (b)Last two rounds

Let P and K be a 128-bit plaintext and a secret key, respectively. Eighteen 64-bit round keys $rk_1, rk_2, \cdots, rk_{18}$ and four 64-bit whitening keys wk_1, wk_2, wk_3, wk_4 are generated from K. Let L_r and R_r $(0 \leq r \leq 18)$ be left and right 64-bit of the internal state in each round. According to the key schedule, the pre-whitening keys are defined as $wk_1||wk_2 = K$. We omit the descriptions of the FL and FL^{-1} layers after the 6^{th} and 12^{th} rounds, since they have no impacts on our work.

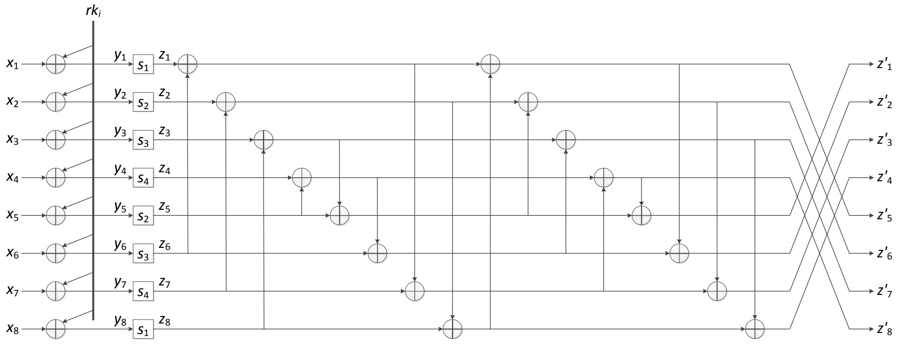


Fig. 11. Camellia SP type F-function

The round function consists of a 64-bit subkey addition, Sbox transformation, and a diffusion layer called P-layer, as shown in Fig.11. Eight parallel 8-bit Sboxes are applied, followed with the P-layer which operates on a 64-bit value $(z_1||z_2||\cdots||z_8)$. The corresponding output $(z'_1||z'_2||\cdots||z'_8)$ is computed as follows.

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

Here, $z[s,t,u,\cdots]$ means $z_s \oplus z_t \oplus z_u \oplus \cdots$

A.2 Attack of Camellia-128

Camellia is a Feistel-SP cipher but its P-layer does not satisfy the maximum branch number. It has the key whitening layer and the whole plaintext is XORed with the whitening key. Our aim is to reveal the master keys through the recovery of the whitening keys.

We use the notations L_i and R_i for Camellia-128 to represent the state immediately after the round operation in the i^{th} round, especially L_0 and R_0 to represent the output of the whitening layer before the first round, respectively. We use the notations PL and PR to differentiate the left and right of plaintext, and S_1, S_2, S_3, S_4, and M to further distinguish the different Sboxes and diffusion operation of Camellia-128. The subscript $[n]$ indicates the n^{th} byte of the state.

Camellia-128 belongs to the case of whitening keys on both branches. To reveal the master key K of Camellia-128, we focus on two 64-bit pre-whitening key wk_1 and wk_2, which are defined as $wk_1||wk_2 = K$. The two whitening keys are XORed with two plaintext blocks PL and PR respectively, before the first round. Hence, our attack target is the first two rounds of Camellia-128. According to the attack procedure in the first round of Camellia-128 which is similar to that of CLEFIA-128, thus we only describe the detailed process to recover wk_1 in the second round.

Now, we show how to recover the whitening keys wk_1 by attacking the round function F of the second round. The whitening keys wk_1 and wk_2 are parallel XORed with two 64-bit plaintext blocks PL and PR respectively, and rk_1 and rk_2 is the round keys as shown in Fig.10. The equivalent key ek_1 and the whitening key wk_2 are recovered in the first round by the attack process. Therefore, we focus on the second round.

According to the specification of Camellia-128 and Equation.1, the 64-bit output L_2 is described by

$$\begin{aligned} L_2 = {} & F(L_1, rk_2) \oplus R_1 \\ = {} & M(S_1(L_{1,[1]} \oplus rk_{2,[1]}), S_2(L_{1,[2]} \oplus rk_{2,[2]}), \\ & \cdots, S_1(L_{1,[8]} \oplus rk_{1,[8]})) \oplus wk_1 \oplus PL. \end{aligned} \tag{11}$$

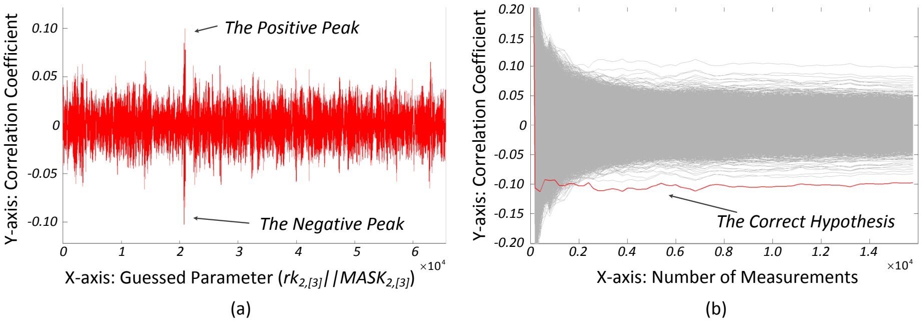
Focusing on the first byte of L_2, Equation.11 could be rewritten as

$$\begin{aligned} L_{2,[1]} &= S_1(L_{1,[1]} \oplus rk_{2,[1]}) \oplus mask_{2,[1]} \oplus wk_{1,[1]} \oplus PL_{[1]} \\ &= S_1(L_{1,[1]} \oplus rk_{2,[1]}) \oplus MASK_{2,[1]} \oplus PL_{[1]}. \end{aligned} \tag{12}$$

where $mask_{2,[1]}$ is generated by $S_2(L_{1,[2]} \oplus rk_{2,[2]}), S_3(L_{1,[3]} \oplus rk_{2,[3]}), \cdots, S_1(L_{1,[8]} \oplus rk_{2,[8]})$ and $MASK_2$ is defined as $MASK_2 = mask_2 \oplus wk_1$.

Before launch attack in the second round, we should calculate the corresponding plaintext message with ek_1 and wk_2 through the inverse transformation of Camellia round function, according to the value of L_1 which meets the chosen message requirements. Fortunately, we find that there is an one-to-one mapping relationship between PR and L_1. Thus, we can control L_2 by enumerating PR, and reveal the value wk_1 by the method as mentioned in Section 3.4. After deriving all the whitening keys, the master key K can be easily derived since $K = wk_1 || wk_2$.

We preset the master key K as four consecutive 32-bit words denoted in hex form, `01234567-89ABCDEF-FEDCBA98-76543210`, in our FPGA implementation with the loop architecture. Thus, it is obvious that $wk_1 =$ `01234567-89ABCDEF`. We use the attack result of the third Sbox (*i.e.*, $rk_{2,[3]}$ and $MASK_{2,[3]}$) as an example. In Fig.12(a), there are two max points `40B8` and `4047` whose correlation coefficients are 0.107 and −0.107 respectively. According to the negative correlation between the power consumption of the FPGA platform and the Hamming distance power model, `4047` is the attack result (*i.e.*, $rk_{2,[3]} =$ `40` and $MASK_{2,[3]} =$ `47`), which is revealed within 4000 power traces as shown in Fig.12(b).



(a)

(b)

Fig. 12. Correlation traces in different (a) hypothesis and (b) number of measurements on Camellia