# A Practical Chosen Message Power Analysis Method on the Feistel-SP ciphers with Applications to CLEFIA and Camellia

Chenyang Tu[1,2,3], Neng Gao[1,2], Zeyi Liu[1,2,3], Lei Wang[1,2], Zongbin Liu[1,2], and Bingke Ma[1,2,3]

State Key Laboratory of Information Security,
Institute of Information Engineering, CAS, Beijing, China[1]
Data Assurance and Communication Security Research Center, CAS, Beijing, China[2]
University of Chinese Academy of Sciences, Beijing, China[3]
{chytu,gaoneng,liuzeyi,lwang,zbliu,bkma}@lois.cn

**Abstract.** The Feistel-SP structure is a commonly adopted structure in symmetric cryptography with many practical instances. Differential power analysis (DPA) has proven to be effective against these ciphers with compact implementations within these years. However, the applications of DPA on Feistel-SP ciphers with loop hardware implementations are more complicated and less evaluated in literature, mainly due to the relatively large size (32-bit or more) of the whole round key which often results in complex relations with the targeted intermediate variable.
In this paper, we propose a practical chosen message power analysis method on Feistel-SP ciphers with loop hardware implementations. The essence of the new method lies in the delicate selection of the plaintext set in a chosen message manner. Thus, the input space of the plaintext in our method is decreased from $2^{32}$ or more to $2^8$ or less, which is suitable for practical power analysis. Moreover, we show that the whitening key at the first or last round can be easily revealed with our method, thus leading to the full exposure of the master key thanks to the relations between whitening keys and the master key in many practical ciphers. In order to further manifest the validity of the new method, we carry extensive experiments on two ISO standardized and widely deployed ciphers CLEFIA and Camellia with loop implementations on FPGA, and the master keys are recovered as expected.

**Keywords:** DPA, chosen message, Feistel-SP, hardware implementation, CLEFIA, Camellia, FPGA

## 1 Introduction

The Feistel network and the substitution-permutation network (SPN) structures, represented by DES and AES, are two main up-level structures in symmetric cryptography. In order to combine the advantages of both structures, such as almost identical encryption/decryption procedures in the Feistel structure and

fast diffusion in the SPN structure, many practical ciphers use well-designed SPN functions as the underlying round functions embedded in the Feistel network, which are often referred to as the Feistel-SP ciphers. Some famous instances of this type are the ISO standardized ciphers CLEFIA [2] and Camellia [3].

DPA has proven to be effective against the Feistel-SP ciphers with software or compact hardware implementations within these years, since Kocher *et al.* [1] first proposed it. Normally, 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 it is highly affected by the specific implementing method. The compact architecture of Feistel-SP cipher usually takes several clock cycles to accomplish a single round computation, such as reusing the single substitution circuit (*e.g.* Sbox) several times as a subloop instead of using their duplications concurrently. Due to the high resource reuse, the compact architecture is often applied to light weight implementations for low-resource ubiquitous computing devices such as smart cards and wireless sensor nodes. In these case, the Feistel-SP ciphers are splitted to several single Sboxes as the general SP ciphers. According to the relatively small size (8-bit or less) of sub-key mixed up in the available intermediate variable, these implementations are compromised by the typical DPA, and the possible input space of random test vector is only $2^8$.

The research of DPA on the compact implementations of these ciphers has obtained numerous achievements. For instance, [4, 5] show the typical power analysis on the compact implementations, and [5, 6, 9, 10] propose several different countermeasures. Especially DPA against CLEFIA on software simulation and smart card implementation are introduced separately in [11] and [12]; it is also shown in [13] that the lightweight FPGA implementation of Camellia under the compact architecture does not resist the DPA attack; in [14], a power analysis on the key-schedule process of the Camellia cipher was presented. Specifically, while many DPA attacks mentioned above work with random messages, chosen messages can further simplify the analysis in compact architecture. For example, fixing part of the input constant can decrease the complexity of the analysis [7]. Moreover, a sequence of adaptively chosen message sets, where each message set depends on the results of the DPA attack done on the prior sets, usually offer more efficiency which has been modeled and analyzed in [8, 18].

Compared to the compact architecture, the loop hardware implementation of Feistel-SP has been widely applied with the higher throughput or less calculation time. With the development of circuit technology in recent years, more and more the loop hardware implementations of Feistel-SP ciphers are applied in the low-resource devices. This architecture following the initial design of the Feistel-SP cipher, defines the round function of Feistel-SP as iterated operation, which means that a single round is computed in one clock cycle. It is believed that such architecture would offer natural DPA countermeasure, due to the available intermediate variable complexly mixed with the relatively large size (32-bit or more) of the whole round key.

**Challenge.** To the best of our knowledge, few DPA attacks exist in literature which make use of the available intermediate variable with large word size (32-bit

or more), such as [19]. However, the method proposed in [19] will not be possible in functions where each bit of the output is dependent on every bit of the input, and this type function is used in the Feistel-SP ciphers. Thus, applying the traditional DPA on the loop hardware implementations of Feistel-SP ciphers remains a difficult challenge mainly due to the following reasons:

- **The original attack point disappears in these scenarios.** In the loop architecture, the intermediate variable of a single substitution circuit resides in a short wire instead of register, since the whole round function including substitution and permutation is performed one time in one clock cycle. Thus, the power consumption of a single Sbox calculation is too low to be detected in such architecture.
- **The possible input space of random test vector is too large to perform practical attack.** Due to the loop architecture, the intermediate result written into register is calculated through the whole round function. The relatively fast diffusion in the round function entangles the intermediate variable bits with all the round key bits. More precisely, the relatively large size (32-bit or more) of the whole round key which is complexly mixed in the targeted intermediate variable, is hard to be splitted into several small fractions. Thus, for recovering the whole round key, the possible input space of random plaintext is at least $2^{32}$, which is unpractical.
- **The key whitening operations seem to contradict the basic requirements of the traditional DPA methodology.** The traditional DPA methodology would only work with the known random message assumption. However, in order to enhance the security properties of the ciphers, the key whitening layers are generally performed before the first round and after the last round, thus hiding the input (output) of the first (last) round from the plaintext (ciphertext).

**Our Contributions.** In this paper, we propose a practical chosen message method to attack the Feistel-SP ciphers with loop implementations by power analysis. Our contributions can be briefly summarized from the following aspects:

- **We propose a practical chosen message power analysis method on Feistel-SP ciphers with loop hardware implementations.** Thus, we significantly decrease the size of guessed parameters from the whole of round key to a pair of 8-bit values, *i.e.* the hypothesis space of the secret value falls from $2^{32}$ or more to $2^{16}$, and the input space of the plaintext is decreased from $2^{32}$ or more to $2^8$, which is suitable for practical power analysis.
- **We show that the additional key whitening layer becomes a new vulnerability in our method.** By guessing the value of "round key X-OR whitening key" as a unity, the attack on the Feistel-SP ciphers with the key whitening layer is similar as the scenario without the key whitening layer. Moreover, the whitening keys can be easily recovered at the first or last round in our method. For many ciphers in practice, the recovery of the

whitening keys commonly indicates direct (partical) exposure of the master key, thus the key recovery process is largely simplified with our method for these specific ciphers.

– **We perform extensive experiments on two ISO standardized ciphers CLEFIA and Camellia with loop FPGA implementations.** And experimental results show that all bits of the master keys in both ciphers can be recovered as expected.

**Structure.** 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. 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 Feistel-SP structure and Key Whitening Layer

The Feistel network is one of the most famous architectures in symmetric cryptography. Besides the classical Feistel network adopted by DES, it also has several derivatives, namely, the unbalanced Feistel network as described in [15], alternating Feistel network as described in [16], and the famous type-1, type-2, and type-3 Feistel networks as described in [17]. Well-known block ciphers utilized the Feistel networks include Skipjack (unbalanced), BEAR/LION (alternating), CAST (type-1), CLEFIA(type-2) and MARS(type-3) respectively.

A typical SP type F-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.

Key whitening is a technique intended to enhance the security of an iterated block cipher by adding key-relevant operations before the first round and after the last round without major changes in the algorithm. It consists of steps that combine the data with portions of the key before the first round and after the last round. The most common form is XORing or adding the whitening key to the plaintext/ciphertext.

### 2.2 Specification of CLEFIA

CLEFIA-128 is a type-2 Feistel-SP cipher proposed at FSE 2007 by Shirai *et al.* [2]. It is standardized by ISO [20] 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.1.

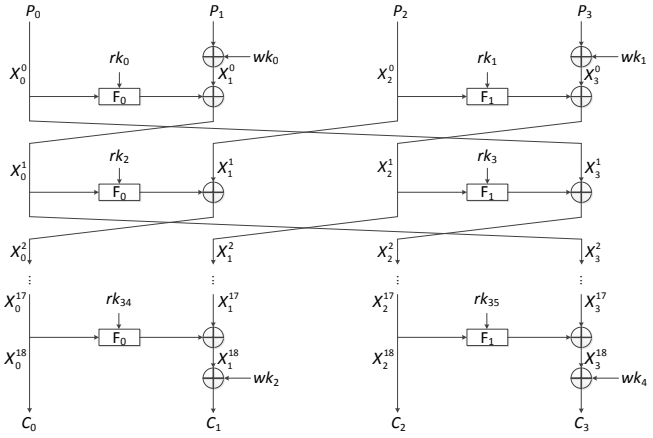


**Fig. 1.** CLEFIA encryption algorithm for 128-bit key

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.

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.2. Four parallel 8-bit Sboxes are applied, followed with an MDS multiplication in each of SP type F-functions. And 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.

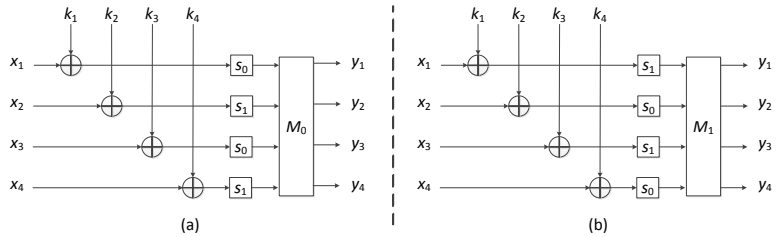


**Fig. 2.** CLEFIA SP type F-functions for (a)$F_0$ (b)$F_1$

### 2.3 Specification of Camellia

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 [21], NESSIE [22], and CRYPTREC [23]. 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.3.

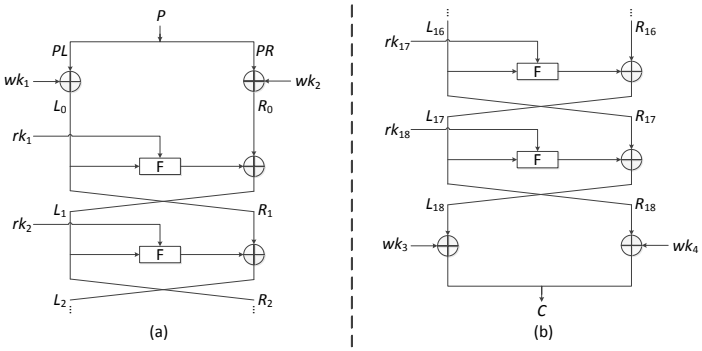


**Fig. 3.** Camellia encryption algorithm for (a)First two rounds (b)Last two rounds

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 two 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.

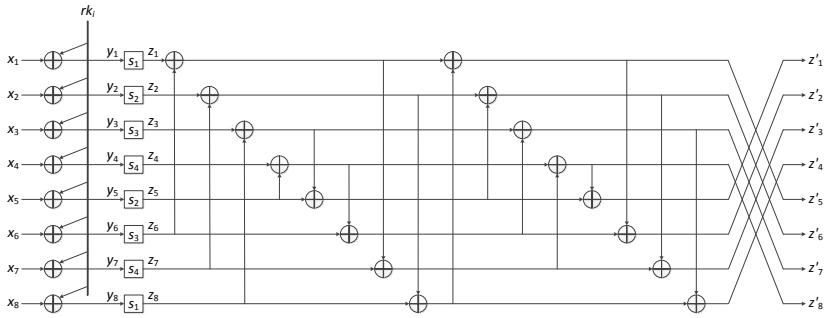


**Fig. 4.** Camellia SP type F-function

The round function consists of a 64-bit subkey addition, Sbox transformation, and a diffusion layer called $P$-layer, as shown in Fig.4. 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. Here, $z[s,t,u,\cdots]$ means $z_s \oplus z_t \oplus z_u \oplus \cdots$

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

## 3 The Chosen Message Power Analysis Method

In this section, we describe the details of our practical chosen message power analysis method. Fristly, we show the potential attack point in loop implementation of Feistel-SP ciphers in DPA attacks, and the difficulty to apply such attacks with existing techniques. Secondly, we propose a chosen message method against the potential attack point to overcome the potential hardness. Then through an elaborate evaluation of the impacts caused by the key whitening layer with our method, we find that the whitening key becomes a new vulnerability. Finally, we present the attack procedures of our method.

### 3.1 The Potential Attack Point in Feistel-SP Loop Scenario

The loop FPGA implementation of Feistel-SP cipher is the investigated scenario as shown in Fig.5. We assume that there are two known parts of the $i^{th}$ round input in Feistel-SP, the left block and the right block denoted by $L_i$ and $R_i$. We also suppose the size of the Sbox is 8-bit. The input of the SP type F-function is defined as $x_1||x_2||\cdots||x_t = R_i$. And Sboxes in the SP type F-function can be distinct and defined as $S_1, S_2, \cdots, S_t$. During the round operation, 8-bit known data $x_i$ is XORed with 8-bit unknown round key $rk_i$ as the Sbox $S_i$ input. The result of $S_i$ is denoted by $y_i$. The result is applied to the permutation $P$ then XORed with $L_i$ and the outcome is denoted by $R_{i+1}$. The output of round operation as the input of the next round is denoted by $L_{i+1}||R_{i+1}$, $s.t.$ $L_{i+1} = R_i$.

More precisely, $R_{i+1}$ would be described by

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

In the scenarios of software and compact hardware implementations, the immediate value $y_i = S_i(x_i \oplus rk_i)$, would be calculated one by one, arise on bus from the MOV instruction or be stored into registers, thus become a DPA weak point. More precisely, when generating $y_i$, we can analyze the subkeys byte by byte because of the little diffusion between the subkeys and $y_i$. In other words, the effective subkey length at this location is only 8-bit, which is too short to resist DPA.

Differing from the above scenarios, no intermediate result except $R_{i+1}$ is written into registers in the loop hardware implementations. Thus, the power consumption of $y_i$ is much less than $R_{i+1}$ and hard to be observed. Therefore, we attack at this point when the data $R_{i+1}$ is being written into registers as shown in Fig.5.

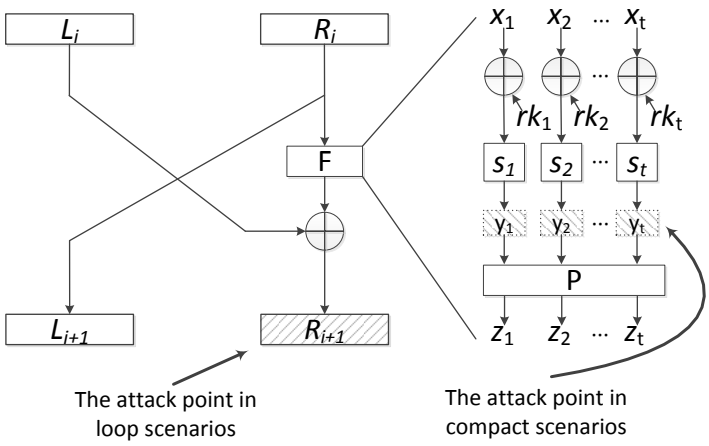


**Fig. 5.** Different DPA attack point of Feistel-SP in different scenarios

However, although we could choose $R_{i+1}$ as the proper attack point, the traditional DPA is not directly applicable due to the relatively large size of the round key $rk$. Due to the fast diffusion of the whole SP type round function, a one-off enumeration of the whole round key seems inevitable with traditional DPA. Compared to the traditional DPA on small scales such as 8-bit Sbox, it is very difficult to cover the input space (*i.e.* $2^{32}$ or more) of the whole round function using random test vectors, due to the large operands. At least one power trace would be measured, according to each random test vector. The

large number of power traces would certainly make the traditional DPA with known random plaintext infeasible.

### 3.2 The Chosen Message Method against $R_{i+1}$

When we choose $R_{i+1}$ as the attack point, the main drawback for traditional DPA is the large size of the subkey, thus our main task is to find an effective way to further decrease the size of the related subkey. Luckily, we achieve this goal, through a chosen message power analysis method, by enumerating all related bits of the plaintext corresponding to the certain subkey while fixing other non-related bits.

Before we further illustrate the details of the method, we would like to clarify several facts about the Feistel-SP ciphers.

**Fact 1.** *The output of Sbox would be a constant value, while a constant plaintext XORed with a constant key is supplied to Sbox.*

**Fact 2.** *As shown in Fig.5, the permutation operation, a linear bijection denoted by $P$, generally uses an MDS multiplication. It implies that there exists a new permutation $P'$ for one byte input of $P$, s.t. $P'_{[i]}(y_i) \oplus mask_i = P_{[i]}(y_1, y_2, \cdots, y_t)$, where $i \in [1, t]$, $(y_1, y_2, \cdots, y_t)$ is the input of $P$, the subscript $[i]$ indicates the $i^{th}$ byte output of one function and the $mask_i$ is a byte variable, which has constant value once all of $y_j$ $(j \neq i)$ are fixed.*

Now, we will focus on the SP type F-function and the attack point $R_{i+1}$. As described in Section 2.1, the SP type F-function is defined as

$$F(R_i, rk) = P(S_1(x_1 \oplus rk_1), S_2(x_2 \oplus rk_2), \cdots, S_t(x_t \oplus rk_t)). \tag{2}$$

Suppose that $x_i$ $(2 \leq i \leq t)$ is fixed, that leads to the constant output of $S_i$. According to the Fact.2 and focusing on the first byte of SP type F-function, Equation.2 would be described by

$$\begin{aligned} F_{[1]}(R_i, rk) &= P_{[1]}(S_1(x_1 \oplus rk_1), S_2(x_2 \oplus rk_2), \cdots, S_t(x_t \oplus rk_t)) \\ &= P'_{[1]}(S_1(x_1 \oplus rk_1)) \oplus mask_1. \end{aligned} \tag{3}$$

with $mask_1$ defined as a byte variable, which has constant value once the $x_2, x_3, \cdots, x_t$ are fixed since the round keys have already been prefixed.

The attack point $R_{i+1}$ is written as $R_{i+1,[1]}||R_{i+1,[2]}||\cdots||R_{i+1,[t]}$, where the subscript $[n]$ indicates the $n^{th}$ byte of the state, and $R_{i+1,[j]}$ is the $j^{th}$ byte variable with $j \in [1, t]$. With Equation.3 and focusing on the first byte of $R_{i+1}$, Equation.1 would be described by

$$\begin{aligned} R_{i+1,[1]} &= F_{[1]}(R_i, rk) \oplus L_{i,[1]} \\ &= P_{[1]}(S_1(x_1 \oplus rk_1), S_2(x_2 \oplus rk_2), \cdots, S_t(x_t \oplus rk_t)) \oplus L_{i,[1]} \\ &= P'_{[1]}(S_1(x_1 \oplus rk_1)) \oplus mask_1 \oplus L_{i,[1]}. \end{aligned} \tag{4}$$

with $L_{i,[1]}||L_{i,[2]}||\cdots||L_{i,[t]} = L_i$.

At this time, $R_{i+1,[1]}$ is highly related to $rk_1$, and $R_{i+1,[2]}, R_{i+1,[3]}, \cdots, R_{i+1,[t]}$ will be treated as noise. Now, we can attack $R_{i+1,[1]}$ by enumerating 8-bit $x_1$ while fixing other bits of $R_i$ and $L_i$, and then building the Hamming Distance power model. Although the possible hypotheses space of both 8-bit $rk_1$ and 8-bit $mask_1$ is $2^{16}$, the possible input space of random test vector $x_1$ is only $2^8$, and the total number of power traces is at least 256, which is a suitable trace-scale for practical DPA. By that analogy, we could analyze $R_{i+1,[2]}, R_{i+1,[3]}, \cdots, R_{i+1,[t]}$ byte by byte in a similar way, and reveal the values of $rk_2, rk_3, \cdots, rk_t$.

### 3.3 The Impacts of Key Whitening Layer

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 plaintexts (ciphertexts). It seems that such operations would increase the size of unknown parameters, and raise the difficulty to launch a DPA attack. Now, we will evaluate how much impact does key whitening layer have on our method.

Suppose that the plaintext message $M$ is XORed with whitening key $WK$ before the Feistel-SP first round, described by

$$\begin{aligned} L_1||R_1 &= M \oplus WK \\ &= (ML \oplus wkL)||(MR \oplus wkR). \end{aligned} \tag{5}$$

with $M = ML||MR$ and $WK = wkL||wkR$.

Equation.2 can be rewritten in byte-wise form:

$$\begin{aligned} F(R_1, rk) = \; & P(S_1(MR_1 \oplus wkR_1 \oplus rk_1), S_2(MR_2 \oplus wkR_2 \oplus rk_2), \\ & \cdots, S_t(MR_t \oplus wkR_t \oplus rk_t)). \end{aligned} \tag{6}$$

where $MR = MR_1||MR_2||\cdots||MR_t$ and $wkR = wkR_1||wkR_2||\cdots||wkR_t$.

And Equation.3 and Equation.4 can be rewritten as

$$\begin{aligned} R_2 = \; & F(R_1, rk) \oplus L_1 \\ = \; & P(S_1(MR_1 \oplus wkR_1 \oplus rk_1), S_2(MR_2 \oplus wkR_2 \oplus rk_2), \\ & \cdots, S_t(MR_t \oplus wkR_t \oplus rk_t)) \oplus wkL \oplus ML. \end{aligned} \tag{7}$$

$$\begin{aligned} R_{2,[1]} = \; & F_{[1]}(R_1, rk) \oplus L_{1,[1]} \\ = \; & P_{[1]}(S_1(MR_1 \oplus wkR_1 \oplus rk_1), S_2(MR_2 \oplus wkR_2 \oplus rk_2), \\ & \cdots, S_t(MR_t \oplus wkR_t \oplus rk_t)) \oplus wkL_1 \oplus ML_1 \\ = \; & P'_{[1]}(S_1(MR_1 \oplus wkR_1 \oplus rk_1)) \oplus mask_1 \oplus wkL_1 \oplus ML_1 \\ = \; & P'_{(8)}(S_1(MR_1 \oplus ek_1)) \oplus MASK_1 \oplus ML_1. \end{aligned} \tag{8}$$

with $ML = ML_1||ML_2||\cdots||ML_t$, $wkL = wkL_1||wkL_2||\cdots||wkL_t$ and the equivalent key $ek = wkR \oplus rk$, the equivalent mask $MASK = mask \oplus wkL$, *s.t.* $ek_i = wkR_i \oplus rk_i$ and $MASK_i = mask_i \oplus wkL_i$.

Now, we can use our method to recover each byte of the equivalent key and the equivalent mask, as $ek_1, ek_2, \cdots, ek_t$ and $MASK_1, MASK_2, \cdots, MASK_t$. According to Fact.1 and Fact.2, we could iteratively calculate each of $mask_i$ by all parts of $ek$. More precisely, $mask_1$ is calculated by $P'_{[1]}(S_1(MR_1 \oplus ek_1)) \oplus P_{[1]}(S_1(MR_1 \oplus ek_1), S_2(MR_2 \oplus ek_2), \cdots, S_t(MR_t \oplus ek_t))$ due to Equation.8, and $mask_j$ ($j \in [2, t]$) is calculated similarly. Then the value of $wkL_i$ could be also iteratively calculated by $wkL_i = MASK_i \oplus mask_i$. The right half of the pre-whitening key $wkR$ and the first round key can be recovered in a similar way at the second round. And the post-whitening key can also be revealed from the decryption direction analogously. Instead of offering strong DPA resistance, the key whitening layer has now become a new vulnerability in our method, since the revealed whitening keys, which normally contain (partial) information of the master key, often largely simplify the recovery process of the master key.

### 3.4 The Attack Procedures of Our Method

Our method is effective against any round of Feistel-SP ciphers with loop hardware implementations. When the target is the first round, we can enumerate all related bits of the plaintext corresponding to the targeted input byte of the first round and fix other non-related bits. In the case of the subsequent rounds, the plaintext set is usually generated by adaptive chosen message method, through running the inversion of encrypted operation with the known previous round keys and the expected input of the targeted round. Fortunately, the whitening keys, which have become new vulnerabilities for many practical ciphers, are usually used before the first round or after the last round. Thus, our method performed only in the first or last round is sufficient to recover the master key in most cases.

For the sake of simplicity, the specific procedures of our chosen message power analysis method performed on the first round are as follows, while the specific procedures to attack other rounds are similar. Also notice that attacking the last few round requires chosen ciphertext set.

**Step 1. Establish the Chosen Plaintext Set.** The first step of our method is to establish the chosen plaintext set. The plaintext set needs to enumerate all attack values of the target bytes, *e.g.* $MR_1$ and fix other bytes to constants.

**Step2. Measure the Power Consumption.** The second step of our method is to measure the power consumption of the Feistel-SP loop hardware implementation. For each of these encryption runs, the plaintext should be one element from the chosen plaintext set.

**Step3. Analyze the Power Traces.** The third step of our method is to analyze the power traces measured in Step2. The analysis strategy would be DPA or CPA, to reveal the corresponding part of the equivalent key and the equivalent mask, *e.g.* $ek_1$ and $MASK_1$.

**Step4. Reveal the Equivalent Key and Equivalent Mask Values.** The next step of our method is to reveal the equivalent key and the equivalent mask values, *e.g.* $ek$ and $MASK$, by repeating Step1 $\sim$ 3.

**Step5. Calculate the Key Value.** After deriving the equivalent key and the

equivalent mask values, the round key or whitening key values would be calculated by the equivalent key and the equivalent mask.

## 4 Applications

We apply these techniques to two typical Feistel-SP ciphers, CLEFIA-128 and Camellia-128, 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.

### 4.1 Application to 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 half of the plaintext blocks are 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.1, 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. 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 gradually. 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 in section 2.2 and Equation.1, the 32-bit output $X_0^1$ is described by

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

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

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

where $mask_{1[1]}$ 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 $MASK_1$ is defined as $MASK_1 = mask_1 \oplus wk_0$.

Therefore, the procedures of our approach against CLEFIA-128 are as follows:

**Step1. Establish the Chosen Plaintext Set.** The first step of our approach is to enumerate all possible values of the target byte $P_{0,[1]}$, and fix other bytes of the plaintext to a randomly chosen constant. For the sake of simplicity, the other bytes are fixed to zero. The chosen plaintext set is shown in Table.1.

**Table 1.** The chosen plaintext set of CLEFIA-128 to analyze $rk_{0,[1]}$

| Test Vector Number | Test Vector Value |
|---|---|
| $Vec_0$ | $0x$00000000_00000000_00000000_00000000 |
| $Vec_1$ | $0x$01000000_00000000_00000000_00000000 |
| $Vec_2$ | $0x$02000000_00000000_00000000_00000000 |
| ... | ... |
| $Vec_{255}$ | $0xFF$000000_00000000_00000000_00000000 |

**Step2. Measure the Power Consumption.** The second step of our approach is to measure the first round power consumption of the CLEFIA-128 loop FPGA implementation. For each of these encryption runs, the plaintext should be one element from the chosen plaintext set.

**Step3. Analyze the Power Traces.** The third step of our approach is to analyze the power traces measured in Step2. The analysis strategy would be CPA, to reveal the first byte of the round key $rk_{0,[1]}$ and the first byte of the equivalent mask $MASK_{1,[1]}$.

**Step4. Reveal the Round Key and Equivalent Mask Values.** The next step of our approach is to reveal each byte of the round key $rk_{0,[2]}$, $rk_{0,[3]}$, and $rk_{0,[4]}$, and the equivalent mask $MASK_{1,[2]}$, $MASK_{1,[3]}$, and $MASK_{1,[4]}$, by repeating Step1 $\sim$ 3.

**Step5. Calculate the Pre-Whitening Key $wk_0$.** After deriving the round key and the equivalent mask values, the values of the corresponding fixed mask bytes $mask_{1,[1]}$, $mask_{1,[2]}$, $mask_{1,[3]}$, and $mask_{1,[4]}$, would be calculated by the round key $rk_0$. Then the first half of pre-whitening key block $wk_0$, can be easily derived with the equation $wk_0 = mask_1 \oplus MASK_1$.

**Step 6. Calculate the Pre-Whitening Key $wk_1$.** The next step of our approach is to reveal the other half of pre-whitening key $wk_1$. It can be easily done by analyzing the F-function $F_1$ in the first round with similar procedures from Step1 to Step5.

**Step 7. Calculate the Post-Whitening Key Values.** The next step of our

approach is to reveal the post-whitening keys $wk_2$ and $wk_3$, by chosen ciphertext power analysis similarly repeating Step1 $\sim$ 6.

**Step 8. Calculate the Master Key $K$.** After deriving all the whitening keys, the master key $K$ can be easily derived since $K = wk_0||wk_1||wk_2||wk_3$.

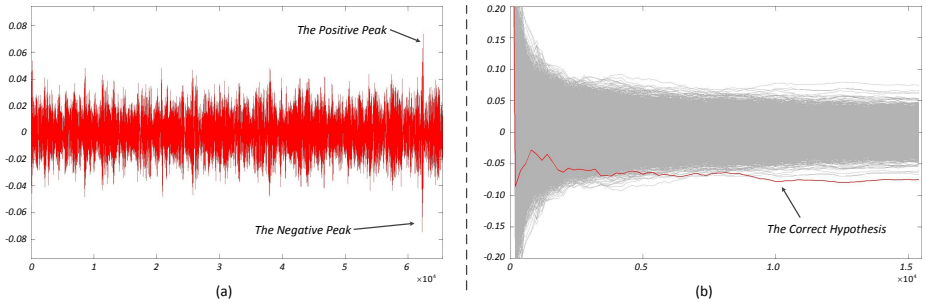


**Fig. 6.** Correlation traces in different hypothesis and different number of measurements

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 demonstrate the attack result of the first Sbox (*i.e.* $rk_{0,[1]}$ and $MASK_{1,[1]}$) as an example. In Fig.6(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.6(b).

We repeat the above procedure 3 times, 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.

### 4.2 Application to 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.

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 $wk_1||wk_2 = K$. We do not need to calculate the inverse transformation of Camellia key schedule to reveal $K$, if we get the whitening key value. 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.

Now, we show how to recover the whitening keys $wk_1$ and $wk_2$ by attacking the round function $F$ of the first two rounds. The whitening keys $wk_1$ and $wk_2$ are parallel XORed with two 64-bit plaintext blocks $PL$ and $PR$ respectively, and $rk_1$ is the round key as shown in Fig.3 . According to the specification in section 2.3 and Equation.1, the 64-bit output $L_1$ is described by

$$\begin{aligned} L_1 = &\ F(L_0, rk_1) \oplus R_0 \\ = &\ M(S_1(PL_{[1]} \oplus wk_{1,[1]} \oplus rk_{1,[1]}), S_2(PL_{[2]} \oplus wk_{1,[2]} \oplus rk_{1,[1]}), \\ &\ \cdots, S_1(PL_{[8]} \oplus wk_{1,[8]} \oplus rk_{1,[8]})) \oplus wk_2 \oplus PR. \end{aligned} \tag{11}$$

Focusing on the first byte of $L_1$, Equation.11 could be rewritten as

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

where $ek_1$ is defined as $ek_1 = wk_1 \oplus rk_1$, $mask_{1,[1]}$ is generated by $S_2(PL_{[2]} \oplus ek_{1,[2]}), S_3(PL_{[3]} \oplus ek_{1,[3]}), \cdots, S_1(PL_{[8]} \oplus ek_{1,[8]})$ and $MASK_1$ is described by $MASK_1 = mask_1 \oplus wk_2$.

In similar way, the 64-bit output $L_2$ is described by

$$\begin{aligned} L_2 = &\ F(L_1, rk_2) \oplus R_1 \\ = &\ M(S_1(L_{1,[1]} \oplus rk_{2,[1]}), S_2(L_{1,[2]} \oplus rk_{2,[2]}), \\ &\ \cdots, S_1(L_{1,[8]} \oplus rk_{1,[8]})) \oplus wk_1 \oplus PL. \end{aligned} \tag{13}$$

Focusing on the first byte of $L_2$, Equation.13 would be described by

$$\begin{aligned} L_{2,[1]} &= S_1(L_{1,[1]} \oplus rk_{2,[1]}) \oplus mask_{2,[1]} \oplus wk_{1,[1]} \oplus PL_{[1]} \\ &= S_1(L_{1,[1]} \oplus rk_{2,[1]}) \oplus MASK_{2,[1]} \oplus PL_{[1]}. \end{aligned} \tag{14}$$

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 $MASK_2 = mask_2 \oplus wk_1$.

Therefore, the steps of our approach against Camellia-128 are as follows:

**Step1. Establish the Chosen Plaintext Set.** The first step of our approach is to enumerate all possible values of the target byte $PL_{[1]}$, and fix other bytes of plaintext $P$ to a randomly chosen constant. For the sake of simplicity, the other bytes are fixed to zero. The chosen plaintext set is shown in Table.2.

**Table 2.** The chosen plaintext set of Camellia-128 to analyze $ek_{1,[1]}$

| Test Vector Number | Test Vector Value |
|---|---|
| $Vec_0$ | $0x$00000000_00000000_00000000_00000000 |
| $Vec_1$ | $0x$01000000_00000000_00000000_00000000 |
| $Vec_2$ | $0x$02000000_00000000_00000000_00000000 |
| ... | ... |
| $Vec_{255}$ | $0xFF$000000_00000000_00000000_00000000 |

**Step2. Measure the Power Consumption.** The second step of our approach is to measure the first round power consumption of the Camellia-128 loop FPGA implementation. For each of these encryption runs,the plaintext should be one element from the chosen plaintext set.
**Step3. Analyze the Power Traces.** The third step of our approach is to analyze the power traces measured in Step2. The analysis strategy would be CPA, to reveal the first byte of the equivalent key $ek_{1,[1]}$ and the first byte of the equivalent mask $MASK_{1,[1]}$.
**Step4. Reveal the Round Key and Equivalent Mask Values.** The next step of our approach is to reveal each byte of the equivalent key $ek_{1,[2]}$, $ek_{1,[3]}$, ..., and $ek_{1,[8]}$, and the equivalent mask $MASK_{1,[2]}$, $MASK_{1,[3]}$, ..., and $MASK_{1,[8]}$, by repeating Step1 $\sim$ 3.
**Step5. Calculate the Pre-Whitening Key $wk_2$.** After deriving the equivalent key and the equivalent mask values, the values of corresponding fixed mask bytes $mask_{1,[1]}$, $mask_{1,[2]}$, ..., and $mask_{1,[8]}$, would be calculated by the equivalent key $ek_1$. Then the half of pre-whitening key $wk_2$, can be easily derived with the equation $wk_2 = mask_1 \oplus MASK_1$.
**Step 6. Calculate the Pre-Whitening Key $wk_1$.** The next step of our approach is to reveal the other half of pre-whitening key $wk_1$. It can be easily done by analyzing the F-function $F$ in the first round with similar procedures from Step1 to Step5.
**Step 7. Calculate the Master Key $K$.** After deriving $wk_1$ and $wk_2$, 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 $wk_2$ = `FEDCBA98-76543210`. We demonstrate the attack result of the fourth Sbox (*i.e.* $ek_{1,[4]}$ and $MASK_{1,[4]}$) as an example. In Fig.7(a), there are two max points `B233` and `B2CC` whose correlation coefficients are 0.075 and $-0.075$ respectively. According to the negative correlation between the power consumption of the FPGA platform and the Hamming distance power model, `B2CC` is the attack result (*i.e.* $ek_{1,[4]}$ = `B2` and $MASK_{1,[4]}$ = `CC`), which is revealed within 10000 power traces as shown in Fig.7(b).

We repeat the above procedure 7 times, and all bytes of $ek_1$ and $MASK_1$ are revealed as $ek_1$ = `AF5286B2-D20D72F2` and $MASK_1$ = `F90216CC-D928E312`. Thus, the value of $mask_1$ is `07DEAC54-8D5EF320`, which is calculated by the round key $ek_1$. And the $wk_2$ is `FEDCBA98-76543210`, calculated by $mask_1 \oplus MASK_1$.

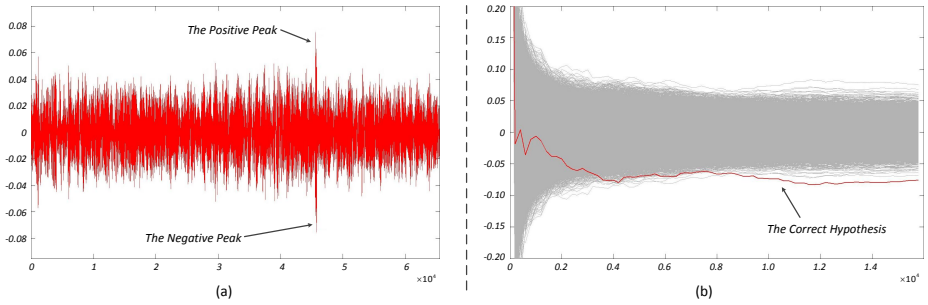


**Fig. 7.** Correlation traces in different hypothesis and different number of measurements

The $wk_1$ are calculated by the similar way with fixing $PL$ and enumerating $PR$ byte by byte. Now, we calculate $K$ is `01234567-89ABCDEF-FEDCBA98-76543210`, which is identical with the preset master key.

## 5 Conclusion and Discussion

In this paper, we propose a practical chosen message power analysis method on Feistel-SP ciphers with loop hardware implementations. We then apply our method to CLEFIA-128 and Camellia-128, and the master keys are recovered as expected. Interestingly, we also find that the whitening key become a new vulnerability in our method. Following the results presented in this work, several problems which are worth further investigations emerge:

- **Optimizations.** The first natural question emerges is whether our method can be further optimized. One possible direction of improvement is taking advantage of the strategy proposed in [8, 18] to discriminate the correct hypothesis from the key candidates in a more efficient way, when launching attack against a specific 8-bit subkey in our approach. 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 [5, 6, 9, 10], can be considered to apply to the loop architecture in order to resist our approach, while the resource consumption will have a corresponding increase. 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.) CRYPTO 1999. LNCS, vol. 1666, pp. 388-397. Springer, Heidelberg (1999).

2. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181-195. Springer, Heidelberg (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.) SAC 2000. LNCS, vol. 2012, pp. 41-54. Springer, Heidelberg (2001).
4. Ors, S.B., Gurkaynak, F.K., Oswald, E., Preneel, B.: Power-Analysis Attack on an ASIC AES Implementation. In: Proccedings International Conference on Information Technology - ITCC 2004, Las Vegas, USA (2004).
5. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks: revealing the secrets of smart cards. Springer, New York (2007). ISBN: 978-0-387-30857-9.
6. Herbst, C., Oswald, E., Mangard, S.: An AES Smart Card Implementation Resistant to Power Analysis Attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239-252. Springer, Heidelberg (2006).
7. Jaffe J.: A first-order DPA attack against AES in counter mode with unknown initial counter. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007, LNCS, vol. 4727, pp. 1-13. Springer, Berlin (2007).
8. 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.) ACM Conference on Computer and Communications Security. ACM, pp. 286-296 (2007).
9. Oswald E., Mangard S., Pramstaller N., Rijmen V.: A side-channel analysis resistant description of the AES S-Box. In: Gilbert, H., Handschuh, H. (eds) FSE, Lecture Notes in Computer Science, vol. 3557, pp. 413-423. Springer, Berlin (2005).
10. Tiri, K., Verbauwhede, I.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: DATE. IEEE Computer Society, pp. 246-251 (2004).
11. Bai, X., Huang, L., Wang, Y.: Differential power analysis attack on CLEFIA block cipher. In: International Conference on Computational Intelligence and Software Engineering, 2009, pp. 1-4.
12. 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.
13. Lu, Y., O'Neill, M.P., McCanny, J.V.: Differential Power Analysis resistance of Camellia and countermeasure strategy on FPGAs. In: FPT 2009, pp: 183-189.
14. Xiao, L., Heys, H.: A Simple Power Analysis Attack against the Key Schedule of the Camellia Block Cipher. In: Information Processing Letters, vol. 95, pp 409-412, Elsevier, 2005.
15. Schneier, B., Kelsey, J.: Unbalanced Feistel networks and block cipher design. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 121-144. Springer, Heidelberg (1996).
16. Anderson, R., Biham, E.: Two practical and provably secure block ciphers: BEAR and LION. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 113-120. Springer, Heidelberg (1996).
17. Zheng, Y., Matsumoto, T., Imai, H.: On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 461-480. Springer, Heidelberg (1990).
18. Veyrat-Charvillon, N., Standaert, F.-X.: Adaptive Chosen-Message Side-Channel Attacks. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 186-199. Springer, Heidelberg (2010).

19. Tunstall, M., Hanley, N., McEvoy, R., Whelan, C., Murphy, C., Marnane, W.: Correlation Power Analysis of Large Word Sizes. In: ISSC 2007, pp. 145-150.
20. ISO/IEC 29192-2:2011. Information technology-Security techniques-Lightweight cryptography-Part 2: Block ciphers (2011).
21. ISO/IEC 18033-3:2010. Information technology-Security techniques-Encryption Algorithms-Part 3: Block ciphers (2010).
22. New European Schemes for Signatures, Integrity, and Encryption(NESSIE). NESSIE PROJECT ANNOUNCES FINAL SELECTION OF CRYPTO ALGORITHMS (2003).
23. Cryptography Research and Evaluation Committees (CRYPTREC). e-Government recommended ciphers list (2003).