Applied Mathematics & Information Sciences
An International Journal

Fault Detection of the Camellia Cipher against Single Byte Differential Fault Analysis

Wei Li[1,2,3], *Xiaoling Xia*[1] *and Yi Wang*[4]

[1] School of Computer Science and Technology, Donghua University, Shanghai 201620, China
[2] Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, Shanghai 200240, China
[3] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[4] Department of Information Science and Technology, East China University of Political Science and Law, Shanghai 201601, China

Received: Jul 8, 2011; Revised Oct. 4, 2011; Accepted Oct. 6, 2011

Abstract: The Camellia is a 128-bit block cipher published by NTT and Mitsubishi in 2000. Since the research of differential fault analysis against Camellia was proposed, much work has been devoted to realizing a more efficient different fault analysis. It is a very strong analysis for ciphers when a single fault is injected into the last several rounds of encryption and the whole secret key could be recovered. Thus, how to detect the faults injected into the Camellia cipher with low overhead of complexity is an open problem. This paper gives an answer to this problem by presenting a fault detection of the Camellia block cipher in the single-byte fault model. Our result in this study could detect the faults with negligible cost when faults are even injected into the last three rounds.

Keywords: Cryptanalysis, Fault Detection, Camellia.

1. Introduction

During the last twenty years a new class of attacks against cryptographic devices has become public. These attacks exploit easily accessible information like power consumption, running time, input–output behavior under malfunctions, and can be mounted by anyone only using low–cost equipment [1-3]. These side–channel attacks amplify and evaluate leaked information with the help of statistical methods, and are often much more powerful than classical cryptanalysis. Examples show that a very small amount of side–channel information is enough to completely break a cryptosystem [4]. While many previously–known cryptanalytic attacks can be analyzed by studying algorithms, the vulnerabilities of side–channel attacks result from electrical behavior of transistors and circuits of an implementation. This ultimately compromises cryptography, and shifts the top priority in cryptography from the further improvement of algorithms to the prevention of such attacks by reducing variations in timing, power and radiation from the hardware, reduction of observability of system behavior after fault injection. Therefore, it extends theoretically the current mathematical models of cryptography to the physical setting which takes into consideration side–channel attacks [5, 6].

As one of side channel attacks, differential fault analysis was first proposed by E. Biham and A. Shamir as an attack on DES in 1997 [7]. The similar attacks have been applied to AES [8-12], ARIA [13] and Camellia [14-16] etc. The DFA attack is based on deriving information about the secret key by examining the differences between a cipher resulting from a correct operation and a cipher of the same initial message resulting from a faulty operation.

As a 128-bit block cipher, Camellia was jointly developed by Nippon Telegraph and Telephone Corporation (NTT) and Mitsubishi Electric Corporation (Mitsubishi) in 2000 [17]. It has now been selected as an international standard by ISO/IEC, and adopted by cryptographic evaluation projects such as NESSIE and CRYPTREC, as well as the standardization activities at IETF. In 2009, Camellia was integrated into the OpenSSL 1.0.0 (beta1) and gradually became one of the most worldwide used block ciphers. Therefore, the strength of Camellia against various cryptanalytic techniques has been analyzed, including differential fault analysis, differential cryptanalysis, linear

* Corresponding author: Xiaoling Xia, e-mail: sherlysha@dhu.edu.cn

cryptanalysis, impossible differential cryptanalysis, collision attack and so on [18-19].

In the literature, some work has been published on the security of Camellia implementations against differential fault analysis [14-16]. These attacks are based on the byte-oriented fault model. They could recover the secret key of Camellia since the error occurs randomly at any position in the last three rounds. To improve the attacking efficiency, the location of fault injection is not same as the location of subkeys which will be recovered. For example, to recover the subkeys in the last round, they induce errors in the penultimate round. This kind of fault injection could derive multiple bytes of one subkey and avoids decreasing the efficiency of fault injection.

In order to resist the above attacks, we propose a fault detection technique to protect Camellia against the previous attacks. Our work not only helps to detect the errors with low overhead of space and time tolerance, but also can be applied in hardware or software implementation. The idea of this attack and the related countermeasure are naturally suitable for other Feistel block ciphers.

The rest of this paper is organized as follows. Section 2 briefly introduces the Camellia cipher. The next two sections propose the differential fault analysis and the previous fault detections. Then section 5 shows our fault detection and simulation on Camellia. Finally section 6 concludes the paper.

2. Description of Camellia

For simplicity, Camellia-128/192/256 are denoted as the three versions of Camellia that use 128, 192 and 256 bits of the secret keys, respectively. Camellia is a d-round Feistel block cipher, where d is 18 for Camellia-128 and 24 for Camellia-192/256 [17]. It has 128-bit XOR operations before the first round and after the last round, called a prewhitening layer and a postwhitening layer, respectively. In every round, the round function F is composed of a nonlinear S-function and a linear P-function. After the 6th and 12th rounds (and 18th round for Camellia-192/256), Camellia has the FL/FL^{-1} function (See Fig. 1). For the rest of this paper, we will use Camellia to the 128-bit secret key version, unless otherwise stated.

2.1. Notations

The following notations are used to describe the Camellia cipher:

Let $X \in (\{0,1\}^8)^{16}$ be the plaintext and $Y \in (\{0,1\}^8)^{16}$ be the ciphertext. Let $k_r \in (\{0,1\}^8)^{16}$ denote the r-th subkey from the secret key K, with $1 \leq r \leq d$. Let $Y^* = (Y_L^*, Y_R^*)$ be the faulty ciphertexts. Let L_{r-1} and R_{r-1} be the left and the right halves of the r-th round input with $1 \leq r \leq d$. Let M_{r-1} be the input of F-function with $1 \leq r \leq d$. Let $|\triangle Y|$, $|\triangle Y_L|$ and $|\triangle Y_R|$ be the number of erroneous bytes in $\triangle Y$, $\triangle Y_L$ and $\triangle Y_R$. The key schedule part generates the subkeys k_r, kl_v and kw_t from the secret key K, where $1 \leq r \leq d$, $1 \leq t \leq 4$ and $1 \leq v \leq 4$ (or $1 \leq v \leq 6$ for Camellia-192/256).

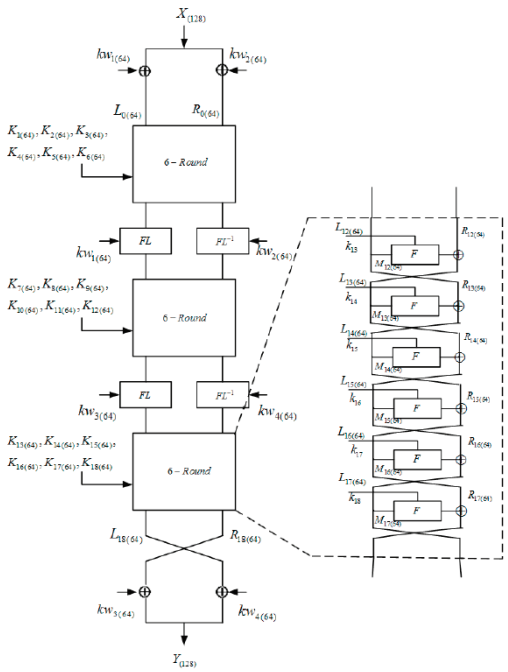

Figure 1 The structure of Camellia.

2.2. Structure

Camellia is composed of three procedures: encryption, decryption and the key schedule. The encryption procedure is described as follows:

Step 1.

$$L_0||R_0 = X \oplus (kw_1||kw_2). \quad (1)$$

Step 2. For $r = 1$ to d do the following:
If $r = 6$ or 12 (or 18 for Camellia–192/256),

$$L_r = R_{r-1} \oplus F(L_{r-1}, k_r), \quad (2)$$
$$R_r = L_{r-1}, \quad (3)$$
$$L_r = FL(L_r, kl_{r/3-1}), \quad (4)$$
$$R_r = FL^{-1}(R_r, kl_{r/3}). \quad (5)$$

else

$$L_r = R_{r-1} \oplus F(L_{r-1}, k_r), \quad (6)$$
$$R_r = L_{r-1}. \quad (7)$$

Step 3.

$$Y = (R_d \oplus kw_3)||(L_d \oplus kw_4). \quad (8)$$

The round function F is defined below:

$$(L_{r-1}, k_r) \mapsto C_r = P(S(L_{r-1} \oplus k_r)), \tag{9}$$

where S and P are defined as follows:

$$S : (F_2^8)^8 \mapsto (F_2^8)^8, \tag{10}$$

$$(a_{1,r}||a_{2,r}||\cdots||a_{8,r}) \mapsto (b_{1,r}||b_{2,r}||\cdots||b_{8,r}), \tag{11}$$

$$b_{1,r} = s_1(a_{1,r}), \tag{12}$$
$$b_{2,r} = s_2(a_{2,r}), \tag{13}$$
$$b_{3,r} = s_3(a_{3,r}), \tag{14}$$
$$b_{4,r} = s_4(a_{4,r}), \tag{15}$$
$$b_{5,r} = s_2(a_{5,r}), \tag{16}$$
$$b_{6,r} = s_3(a_{6,r}), \tag{17}$$
$$b_{7,r} = s_4(a_{7,r}), \tag{18}$$
$$b_{8,r} = s_1(a_{8,r}), \tag{19}$$

where s_1, s_2, s_3 and s_4 are the 8×8 boxes.

$$P : (F_2^8)^8 \mapsto (F_2^8)^8, \tag{20}$$
$$(b_{1,r}||b_{2,r}||\cdots||b_{8,r}) \mapsto (c_{1,r}||c_{2,r}||\cdots||c_{8,r}), \tag{21}$$

where

$$c_{1,r} = b_{1,r} \oplus b_{3,r} \oplus b_{4,r} \oplus b_{6,r} \oplus b_{7,r} \oplus b_{8,r}, \tag{22}$$
$$c_{2,r} = b_{1,r} \oplus b_{2,r} \oplus b_{4,r} \oplus b_{5,r} \oplus b_{7,r} \oplus b_{8,r}, \tag{23}$$
$$c_{3,r} = b_{1,r} \oplus b_{2,r} \oplus b_{3,r} \oplus b_{5,r} \oplus b_{6,r} \oplus b_{8,r}, \tag{24}$$
$$c_{4,r} = b_{2,r} \oplus b_{3,r} \oplus b_{4,r} \oplus b_{5,r} \oplus b_{6,r} \oplus b_{7,r}, \tag{25}$$
$$c_{5,r} = b_{1,r} \oplus b_{2,r} \oplus b_{6,r} \oplus b_{7,r} \oplus b_{8,r}, \tag{26}$$
$$c_{6,r} = b_{2,r} \oplus b_{3,r} \oplus b_{5,r} \oplus b_{7,r} \oplus b_{8,r}, \tag{27}$$
$$c_{7,r} = b_{3,r} \oplus b_{4,r} \oplus b_{5,r} \oplus b_{6,r} \oplus b_{8,r}, \tag{28}$$
$$c_{8,r} = b_{1,r} \oplus b_{4,r} \oplus b_{5,r} \oplus b_{6,r} \oplus b_{7,r}. \tag{29}$$

The two functions FL and FL^{-1} are described in [17].

3. Differential fault analysis on Camellia

3.1. Fault model and fault assumption

The fault model includes the following two assumptions: the attacker has the capability to choose one plaintext to encrypt and obtain the corresponding right and faulty ciphertexts(Chosen Plaintext Attack, CPA); the attacker can induce a single byte error to one layer. However, the location of this byte in this layer and the value of the error are both unknown. As for the attack, we analyze a fault occurring near the end of the algorithm and assume the general random fault model where the fault modifies the processed data in a random way. The adversary does not need to know a priori the random value the fault imposed on the data. As will be shown later, the assumption of fault induction can be relaxed to a certain extend.

3.2. Basic procedure

The basic procedure of this attack is as follows: the right ciphertext is obtained when a plaintext is encrypted with a secret key. We induce a random error in some round of the encryption, and thus obtain a faulty ciphertext. By differential fault analysis, the XOR value of the last subkey can be recovered. Then we could decrypt the right ciphertext to obtain the input of the last round, which is the output of the penultimate round. At last we repeat the above procedure to deduce more related values about subkeys until the secret key is obtained by the key schedule.

4. Differential fault detections on Camellia

Countermeasures against fault attacks could help a cryptographic algorithm to avoid, detect or correct faults. In practice, many proposed schemes are based on fault detection, including code-based technique and redundancy-based technique [20-26].

4.1. Code-based technique

Code based detections are divided into coding method and error detection code (EDC). Coding method means encoding message before encryption and checking errors after decryption. Its overhead depends on encoding and decoding progress to translate plaintexts and ciphertexts into codes. Its time redundancy also depends on the code processes. As for block ciphers, the EDC approach is often used in each rounds' inner parts with the implementation of parity-based EDC. The parity of linear layers is easy to implement since permutations do not change the parity. More consideration should be given to the nonlinear layers. Whether the parity of input joins in encryption determines how the parity constructs. Approximately, 10%$\sim$20% overhead is required, and so does time tolerance.

4.2. Redundancy-based technique

The redundancy-based solution for implementing fault detection in the encryption module is to perform a test decryption immediately after the encryption, and then check whether the original data block is obtained. If a decryption module is already present in the implementation, the hardware overhead reduces to the cost of a comparator for two data blocks of 128 bits. Otherwise, the overhead is close to 100 percents since the decryption module is very similar to the encryption one. The overall time penalty in either of these two cases is the time required to decrypt a data block, plus the time required for the comparison. This technique is independent of the adopted fault model.

The above two techniques of fault detection seem to ensure a high level of security. However, only checking

the correctness of the computed encryption result may not be enough to prevent fault analysis since an attacker may destroy the detector.

5. Our proposed fault detection on Camellia

The previous differential fault analysis on the security of Camellia adopts the basic assumption and fault model as follows:

(1)The attacker has the capability to obtain the right and the corresponding faulty ciphertexts when encrypting one plaintext with the same secret key.
(2)The attacker can induce a single byte error to a 32–bit register. However, the location of this byte in this register and the value of the error are both unknown.

On the above basic assumptions, they induce a random error in the last three rounds, and thus obtain a faulty ciphertext. By differential fault analysis, part bytes of the subkeys in the last round can be recovered. Repeat this procedure until the subkey is obtained. Then they decrypt the right ciphertext to obtain the input of the last round, which is the output of the penultimate round. Repeat the above procedure until the secret key is obtained by the key schedule.

Our objective is to develop fault detection techniques which will be independent of the particular hardware implementation. A fault injected into the first round is comparable to encoding a different input. The injection of a fault in one of the inner rounds is more complicated and it is necessary to follow the errors as they propagate along the execution path. Every round of Camellia consists of the round function, which is composed of three transformations: Subkey exclusive OR, Substitution and Permutation. The propagation of a single fault is influenced by the execution of the round components. The result can be classified into two cases: the fault spreads considerably or the fault affects only one byte in the output. The latter situation includes the Subkey exclusive OR and Substitution transformations, where the error is only moved within a byte, respectively. The remaining permutation is more complex and will therefore greatly influence the propagation of errors.

When using a specific input and injecting a single-byte fault into every different rounds, the number of erroneous bytes in the ciphertext has the following characteristic (See Figure 2):

(1)If all bytes are erroneous, the fault could occur before the last four rounds.
(2)If there are less than 16 nonzero erroneous bytes in average, the fault could occur in the last three rounds. The average number of erroneous bytes is 15.12, 11.16 and 4.33, respectively.

To date, little research has been done on the related attacking method when the faults are induced before the three rounds. Thus, Camellia is secure even if the errors have been induced before the four rounds. We put emphasis on the research of the errors induced into the last three rounds. In the DFA analysis, the attacker must capture at least two ciphertexts, including one right ciphertext and one faulty ciphertext. On the basis of this assumption, we propose a fault detection technique to infer whether the attacker induce faults into the encryption module.

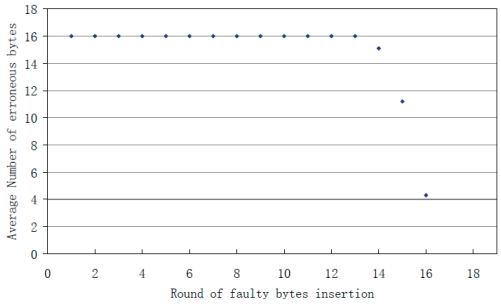

Figure 2 Erroneous bytes in the ciphertext of Camellia.

Table 1 The relationship between some patterns and ciphertext pairs.

Pattern		A ciphertext pair
$\lvert\triangle Y\rvert$	$\lvert\triangle Y_L\rvert, \lvert\triangle Y_R\rvert$	
1	01	$(Y, Y^*), (Y^*, Y^*)$
2	02	(Y^*, Y^*)
5	05	$(Y, Y^*), (Y^*, Y^*)$
6	06, 15	$(Y, Y^*), (Y^*, Y^*)$
7	16, 07	$(Y, Y^*), (Y^*, Y^*)$
8	08, 17, 26	(Y^*, Y^*)
9	18, 27	(Y^*, Y^*)
10	28	(Y^*, Y^*)
13	58	(Y, Y^*)
14	68	(Y^*, Y^*)
15	78	(Y^*, Y^*)
16	88	$(Y, Y^*), (Y^*, Y^*)$

On the byte-oriented fault model in the previous attacks, the fault could be injected into the last three rounds. The pattern is defined within the bounds of remote possibility as the result of the XOR operation between two right ciphertexts (See Table 1). There are two possibilities for a ciphertext pair: a correct ciphertext and a faulty ciphertext (Y, Y^*), two faulty ciphertexts (Y^*, Y^*). If the distribution of a ciphertext difference satisfies these patterns, we could derive that the attacker has induced faults into the encryption module and at least one ciphertext is faulty. In other words, if the ciphertext difference satisfies the distribution of some patterns in Table 1, it shows that the error has been induced into the encryption module. Otherwise, it is not feasible for DFA to derive the secret key of Camellia.

Table 2 Patterns of two faulty ciphertexts.

	01	05	06	15	16	58	68	88
01	01, 02	05,06	06, 07	15, 16	16, 17	58	68	88
05	05, 06	05-08	06-08	15-18	16-18	58	68	88
06	06, 07	06-08	06-08	15-18	16-18	58	68	88
15	15, 16	15-18	16-18	15-18, 25-28	16-18, 26-28	58, 68	68, 78	88
16	16, 17	16-18	16-18	16-18, 26-28	16-18, 26-28	58, 68	68, 78	88
58	58	58	58	58,68	58,68	58-88	68-88	88
68	68	68	68	68,78	68,78	68-88	68-88	88
88	88	88	88	88	88	88	88	88

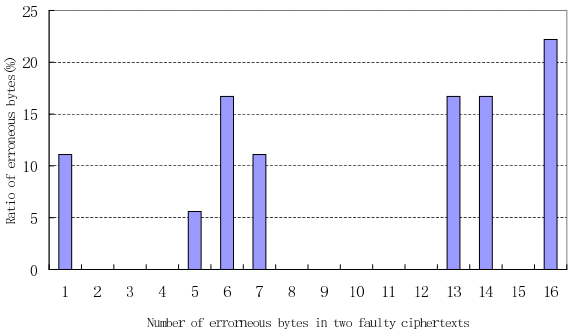

Figure 3 Ration of erroneous bytes in one correct ciphertext and one faulty ciphertext in the last three rounds.

5.1. Patterns of a correct ciphertext and a faulty ciphertext

By the ciphertext difference, we could detect the fault location as Table 1 shows. Some patterns are defined within the bounds of average possibility as the result of the XOR operation between two right ciphertexts in the last three rounds For example, when the error is injected into the last three round, the ratio of 1, 5, 6, 7, 13, 14 and 16 erroneous bytes occur at 11.1%, 5.6%, 16.7%, 11.1%, 16.7%, 16.7%, and 22.2%, respectively. (See Figure 3).

5.2. Patterns of two faulty ciphertexts

In real application, one correct ciphertext and one faulty ciphertext as a ciphertext pair is ideal. However, there exist two faulty ciphertexts. On the basis of Table 2, we build up the pattern of ciphertext difference between two faulty ciphertexts in Table 3. Thus, we derive the relationship between the pattern of two faulty ciphertexts and the fault locations in Table 4. when two errors are injected into the last three round, the ratio of 1, 2, 5, 6, 7, 8, 9, 10, 13, 14, 15 and 16 erroneous bytes occur at 0.76%, 0.76%, 1.89%, 8.12%, 14.41%, 8.81%, 5.41%, 0.94%, 12.46%, 16.99%, 4.91% and 24.54%, respectively (See Figure 4).

Table 3 The relationship between pattern of two faulty ciphertexts and fault injection.

Pattern		Location of fault injections
$\lvert\triangle Y\rvert$	$\lvert\triangle Y_L\rvert, \lvert\triangle Y_R\rvert$	
1	01	$R_{17}R_{17}$
2	02	$R_{17}R_{17}$
5	05	$R_{17}M_{17}$, $M_{17}M_{17}$
6	06	$R_{17}M_{17}$, $R_{17}M_{17}$, $R_{17}L_{17}$, $M_{17}M_{17}$
	15	$R_{17}L_{17}$, $R_{17}M_{16}$, $M_{17}L_{17}$, $M_{17}M_{16}$, $L_{17}L_{17}$, $M_{16}L_{17}$, $M_{16}M_{16}$
7	07	$R_{17}M_{17}$
	16	$R_{17}L_{17}$, $R_{17}M_{16}$, $M_{17}L_{17}$, $M_{17}M_{16}$, $L_{17}L_{17}$, $M_{16}L_{17}$, $M_{16}M_{16}$
8	08	$M_{17}M_{17}$
	17	$R_{17}L_{17}$, $R_{17}M_{16}$, $M_{17}L_{17}$, $M_{17}M_{16}$, $L_{17}L_{17}$, $M_{16}L_{17}$, $M_{16}M_{16}$
	26	$L_{17}L_{17}$, $M_{16}L_{17}$, $M_{16}M_{16}$
9	18	$M_{17}L_{17}$, $M_{17}M_{16}$, $L_{17}L_{17}$, $M_{16}L_{17}$, $M_{16}M_{16}$
	27	$L_{17}L_{17}$, $M_{16}L_{17}$, $M_{16}M_{16}$
10	28	$L_{17}L_{17}$, $M_{16}L_{17}$, $M_{16}L_{16}$
13	68	$R_{17}L_{16}$, $R_{17}M_{16}$, $R_{17}R_{15}$, $M_{17}L_{16}$, $M_{17}M_{16}$, $M_{17}R_{15}$, $L_{17}L_{16}$, $L_{17}M_{16}$, $L_{17}R_{15}$, $M_{16}L_{16}$, $M_{16}L_{17}$, $M_{16}R_{15}$, $L_{16}L_{16}$, $L_{16}M_{16}$, $L_{16}R_{15}$, $M_{16}M_{16}$, $M_{16}R_{15}$, $R_{15}R_{15}$
14	68	$R_{17}L_{16}$, $R_{17}M_{16}$, $R_{17}R_{15}$, $M_{17}L_{16}$, $M_{17}M_{16}$, $M_{17}R_{15}$, $L_{17}L_{16}$, $L_{17}M_{16}$, $L_{17}R_{15}$, $M_{16}L_{16}$, $M_{16}L_{17}$, $M_{16}R_{15}$, $L_{16}L_{16}$, $L_{16}M_{16}$, $L_{16}R_{15}$, $M_{16}M_{16}$, $M_{16}R_{15}$, $R_{15}R_{15}$
15	78	$L_{17}L_{16}$, $L_{17}M_{16}$, $L_{17}R_{15}$, $M_{16}L_{16}$, $M_{16}L_{17}$, $M_{16}R_{15}$, $L_{16}L_{16}$, $L_{16}M_{16}$, $L_{16}R_{15}$, $M_{16}M_{16}$, $M_{16}R_{15}$, $R_{15}R_{15}$
16	2222	

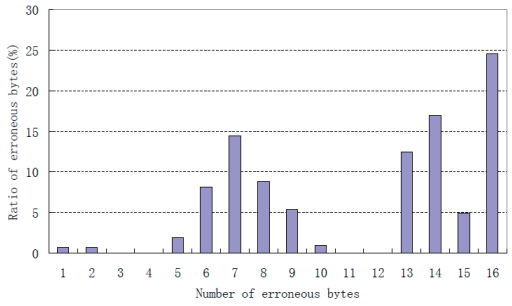

Figure 4 Ration of erroenous bytes of two faulty ciphertexts in the last three rounds.

5.3. Simulations

We implemented our attack on a PC using Visual C++ on a 1.60 GHz centrino with 8 GB memory. The fault induction was simulated by computer software. In this situation, we ran the attack algorithm to 100 encryption unit with different random generated keys. The success rate of our method can detect fault is 77.8% to recover the secret key. The time to complete the attack is less than 1 second. Compared with the previous techniques, the overhead and time tolerance of required for the comparison in our method is negligible. As one countermeasure of Camellia against D-FA, the proposed technique could not only help to detect the errors with low overhead of space and time tolerance, but also be applied in hardware or software implementation.

6. Conclusion

In this paper, we examine the security analysis of Camellia with its fault injection simulation in implementation. It is simple to detect errors in real applications, and provides a practical approach for fault detection on other block ciphers.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No. 61003278, the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, the open research fund of State Key Laboratory of Information Security and the Fundamental Research Funds for the Central Universities.

References

[1] M. Tian, L. Huang and W. Yang, Appl. Math. Inf. Sci, **6**, 419 (2012).
[2] G.. Zhang and M. Xie, Appl. Math. Inf. Sci, **5**, 219 (2011).
[3] A. Banerjee and A. Pathak. Appl. Math. Inf. Sci, **6**, 157 (2012).
[4] T. Zhukabayeva, O. Sembiyev and V. Khu. Appl. Math. Inf. Sci, **6**, 789 (2012).
[5] Y. Lee, Appl. Math. Inf. Sci, **6**, 425 (2012).
[6] Y. Hung, Appl. Math. Inf. Sci, **7**, 431 (2013).
[7] E. Biham and A. Shamir, LNCS **1294**, 513 (1997).
[8] J. Blomer and J. P. Seifert, LNCS **2742**, 162 (2003).
[9] P. Dusart, G. Letourneux and O. Vivolo, LNCS **2846**, 293 (2003).
[10] C. Giraud, LNCS **3373**, 27 (2005).
[11] P. Gilles and J. J. Quisquater, LNCS **2779**, 77 (2003).
[12] M. Amir, T. M. S. Mohammad and S. Mahmoud, LNCS **4249**, 91 (2006).
[13] W. Li, D. Gu and J. Li, Information Sciences, **178**, 3727 (2008).
[14] W. Li, D. Gu, J. Li, Z. Liu and Y. Liu, Journal of Systems and Software, **83**, 844 (2010).
[15] Y. Zhou, W. Wu, N. Xu and D, Feng, Chinese Journal of Electronics, **18**, 1 (2009).
[16] X. Zhao and T. Wang, http://eprint.iacr.org/2010/026.pdf.
[17] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima and T. Tokita, LNCS **2012**, 39 (2000).
[18] L. Duo, C. Li and K. Feng, LNCS **3897**, 51 (2005).
[19] Y. Hatano, H. Sekine and T. Kaneko, LNCS **2595**, 129 (2003).
[20] M. Karpovsky, K. J. Kulikowski and A. Taubin, International Conference on Dependable Systems and Networks, 19 (2004).
[21] R. Karri, K. Wu, P. Mishra and Y. Kim, IEEE Transactions on Computer-Aided Design, **21**, 1509 (2002).
[22] T. G. Malkin, F. X. Standaert and M. Yung, LNCS, **4236**, 159 (2006).
[23] K. Wu, R. Karri, G. Kuznetsov and M. Goessel, International Test Conference, 1242 (2004).
[24] N. Joshi and K. Wu, LNCS, **3156**, 400 (2004).
[25] R. Karri and M. Gossel, International Test Conference, 919 (2003).
[26] R. Karri, G. Kuznetsov and M. Gossel, LNCS, **2779**, 113 (2003).

Wei Li is a associate professor at School of Computer Science and Technology in Donghua University. Her was awarded her B.S. degree in Engineering from Anhui University in 2002, and M.S. degree and Ph.D. degree in Engineering from Shanghai Jiao Tong University in 2006 and 2009, respectively. She is a members of IEEE and ACM. Her main research interests focus on applied cryptography and computer security.

Xiaoling Xia is a associate professor at School of Computer Science and Technology in Donghua University. Her was awarded her B.S. degree, M.S. degree and Ph.D. degree in Engineering from Shanghai Jiao Tong University in 1988, 1992 and 1994, respectively. Her main research interests focus on computer security.

Yi Wang is currently an associate professor in Department of Information Science and Technology, East China University of Political Science and Law. She was awarded her Ph.D. degree from Shanghai Jiao Tong University in 2004. Her research interests include information and network security.