International Journal of Computational Intelligence Systems, Vol. 5, No. 5 (September, 2012), 896-904

Single Byte Differential Fault Analysis on the LED Lightweight Cipher in the Wireless Sensor Network

Wei Li
School of Computer Science and Technology, Donghua University
Shanghai, 201620, China
Shanghai Key Laboratory of Integrate Administration Technologies for Information Security
Shanghai, 200240, China
State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences
Beijing 100093, China

Da-wu Gu
Department of Computer Science and Engineering, Shanghai Jiao Tong University
Shanghai, 200240, China

Xiao-ling Xia**, *Chen Zhao
School of Computer Science and Technology, Donghua University
Shanghai, 201620, China
E-mail: sherlysha@dhu.edu.cn

Zhi-qiang Liu, Ya Liu
Department of Computer Science and Engineering, Shanghai Jiao Tong University
Shanghai, 200240, China

Qing-ju Wang
Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven
Heverlee, B-3001, Belgium

Received 15 December 2011

Accepted 15 June 2012

Abstract

The LED is a new lightweight cipher, which was published in CHES 2011. This cipher could be applied in the Wireless Sensor Network to provide security. On the basis of the single byte-oriented fault model, we propose a differential fault analysis on the LED cipher. The attack could recover its 64-bit secret key by introducing 4 faulty ciphertexts, and 128-bit secret key by introducing 8 faulty ciphertexts. The results in this study will be beneficial to the analysis of the same type of other iterated lightweight ciphers.

Keywords: Wireless Sensor Network; Lightweight Cipher; LED; Differential Fault Analysis.

1. Introduction

The LED is a new Substitution-Permutation Network (SPN) lightweight cipher published in CHES 2011[1]. It has a good compact hardware implementation and

*Corresponding Author: Xiaoling Xia.

software-friendly features, and thus could be applied in the sensor network, RIFD tag deployment and other application characterized by highly-constrained devices. The strength of the LED cipher against various classical cryptanalysis has been analyzed, including differential cryptanalysis, linear cryptanalysis, algebraic attack and so on[1].

Other than classical cryptanalysis, differential fault analysis (DFA) is a new class of cryptanalysis on cryptographic devices. It was first proposed by E. Biham and A. Shamir on DES[2] in 1997. The similar attacks have been applied to AES[3-8], Triple-DES[9], RC4[10], Camellia[11], ARIA[12], SMS4[13-14], PRESENT[15] and so on. The DFA exploits easily accessible information like input-output behavior under malfunctions, amplifies and evaluates the leaked information with the help of mathematical methods. It is based on deriving information about the secret key by examining the differences between a cipher resulting from correct operation and a cipher of the same initial message resulting from faulty operation. This analysis is often much more powerful than classical cryptanalysis.

As for the SPN block ciphers, there are two types of DFA methods, which relate to the recovery of subkeys:

(i) The bit number of the fault is less than or equals to that of the input of an S-box. For example, the fault is one bit or byte, while the input of an S-box is one byte. In this case, one fault's diffusion affects at most one S-box in the inducing round, and the fault propagation path is simple. Thus, it is not difficult for the attackers to derive the relationship between the subkeys and the ciphertext difference. Many studies have been conducted on the security of these SPN block ciphers against the DFA[3-8, 12, 15].

(ii) The bit number of the fault is greater than that of the input of an S-box. For example, the fault is a random byte, while the input of an S-box is 4 bits. Thus, the fault propagation paths may intersect each other. Thus it is not easy to deduce the relationship between the secret key and the ciphertext difference. For example, The LED is such a representative SPN block cipher which chooses 4 bits as a processing unit and its input of an S-box has 4 bits for the high efficiency in software and hardware. In the real circumstances, it is usual for the DFA attacker to choose the single-byte oriented fault model to deduce the subkeys and the secret key for its easy implementation and general applications. However, few studies have been done on the single byte differential fault analysis on the LED cipher.

In this study, we propose a single byte differential fault analysis method to recover the secret key of LED. The method could induce one-byte errors into the encryption. Both the locations and the values of the errors are unknown. By retrieving the related values of subkeys, our method requires 4 and 8 ciphertexts to recover the 64-bit and 128-bit secret keys of LED, respectively.

The rest of this paper is organized as follows. Section 2 briefly introduces the LED cipher. The next section describes the basic assumption and basic idea of DFA. Then section 4 proposes our DFA analysis to recover the secret key. Section 5 summarizes the attacking complexity. Section 6 shows the experimental results of the DFA on LED. Finally section 7 concludes the paper.

2. Description of the LED Cipher

The LED is a 64-bit SPN block cipher with two primary instances taking 64-bit and 128-bit secret keys. It has l rounds, which is 32 for LED-64 and 48 for LED-128 as Fig. 1 shows. The cipher is composed of encryption, decryption and the key schedule.

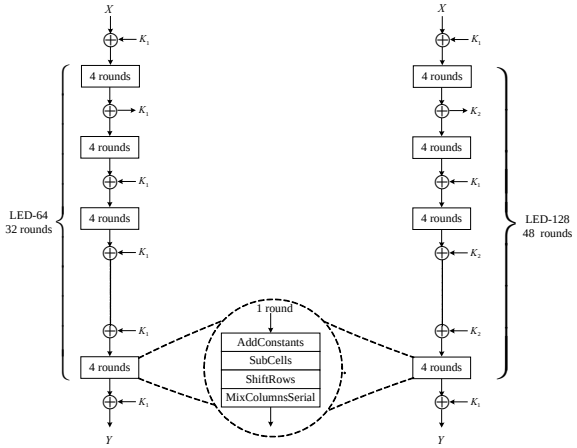

Fig. 1. The structure of LED-64 and LED-128.

2.1. Encryption

The LED could be pictured as a rectangular array of 4 bits, having four rows and four columns. Let X be the plaintext and Y be the ciphertext. Let K_1 and K_2 denote the subkeys from the secret key K. The intermediate result is denoted as $STATE \in (\{0,1\}^4)^{16}$. In every four rounds, the $STATE$ is XORed with a 64-bit subkey,

called *AddRoundKey.* The basic component of LED is a sequence of four identical rounds, and each round includes *AddConstants*, *SubCells*, *ShiftRows* and *MixColumnsSerial* in sequence:

(i) *AddConstants* is the transformation that processes the *STATE* with a constant.

(ii) *SubCells* is the transformation that processes a *STATE* with a nonlinear 4-bit substitution table (S-box).

(iii) *ShiftRows* is the transformation that cyclically shifts the last three rows of the *STATE* by different offsets.

(iv) *MixColumnsSerials* is the transformation that takes all the columns of the *STATE* and mixes their data to produce new columns.

2.2. *Decryption*

The decryption is the same as that of encryption, including the subkeys with the same order.

2.3. *Key Schedule*

The secret key K is the input of a key schedule to produce the subkeys for each round. In LED-64, the relationship between the secret key K and the subkey K_1 is as follows:

$$K = K_1. \quad (1)$$

In LED-128, the relationship among the secret key K, the subkeys K_1 and K_2 is as follows:

$$K = K_1 \| K_2. \quad (2)$$

3. Basic Idea of Our Attack

3.1. *The Basic Assumption*

The DFA analysis exploits the difference between a normal and a faulty ciphertext stemming from encryptions of the same plaintext. Our proposed fault model includes the following two assumptions:

(i) The attacker has the capability to choose one plaintext to encrypt and obtain the corresponding right and faulty ciphertexts (Chosen Plaintext Attack, CPA).

(ii) The attacker could induce a single byte error to one transformation. However, the location of this byte in this round and the value of the error are both unknown.

3.2. *The Basic Idea*

The main procedure of this attack is as follows:

(i) The right ciphertext is obtained when a plaintext is encrypted with a secret key.

(ii) We induce a random error in the *l-2*-th round of the encryption, and thus obtain a faulty ciphertext. By differential analysis, the value of the last subkey K_1 can be recovered. If the key size is 64 bits, then jump step (iv); else jump step (iii).

(iii) A random error is induced in the *l-6*-th round of the encryption, we could decrypt the right ciphertext by the subkey K_1 to obtain the input of the last four round, which is the output of the fifth round from the end. Repeat the same procedure to deduce the subkey K_2.

(iv) The secret key K could be deduced on the basis of the key schedule.

4. Single Byte Differential Fault Analysis on LED

4.1. *Notations*

The following notations are used to describe the LED and its analysis.

Let $X \in (\{0,1\}^4)^{16}$ be the plaintext and $Y \in (\{0,1\}^4)^{16}$ be the ciphertext. Let $K_1 \in (\{0,1\}^4)^{16}$ and $K_2 \in (\{0,1\}^4)^{16}$ denote the subkeys from the secret key $K \in (\{0,1\}^4)^{16}$.

Let AC_j^d, SC_j^d, SR_j^d and MC_j^d represent the j-th 4-bit output value of the *AddConstants*, *SubCells*, *ShiftRows*, and *MixColumnsSerial* layers in the d-th round with $1 \le d \le l$ and $0 \le j \le 15$, respectively.

Let AR_j^c represents the j-th 4-bit output value of the *AddRoundkey* layer in the c-th 4 round with $1 \le c \le \lceil l/4 \rceil$ and $0 \le j \le 15$.

Let ΔAC_j^d, ΔSC_j^d, ΔSR_j^d and ΔMC_j^d represent the j-th 4-bit output difference of the *AddConstants*, *SubCell*, *ShiftRows*, and *MixColumnsSerial* layers in the d-th round with $1 \le d \le l$ and $0 \le j \le 15$, respectively.

Let ΔAR_j^c represents the j-th 4-bit output difference of the *AddRoundkey* layer in the c-th 4 round with $1 \le c \le \lceil l/4 \rceil$ and $0 \le j \le 15$.

The relationship between the input difference and output difference in the *SubCells* transformation is defined as follows:

$$IN(\Delta AC_j^d, \Delta SC_j^d) = \{AC_j^d \mid AC_j^d \in \{0,1\}^4, S(AC_j^d) \oplus S(AC_j^d \oplus \Delta AC_j^d) = \Delta SC_j^d\}, 0 \le j \le 15, 1 \le d \le l\}, \quad (3)$$

where the S represents a 4×4 substitution in the *SubCells layer.*

Let $AddConstants^{-1}$, $SubCells^{-1}$, $ShiftRows^{-1}$, and $MixColumnsSerial^{-1}$ represent the inverse operations of the *AddConstants*, *SubCells*, *ShiftRows*, and *MixColumnsSerial* transformations, respectively.

4.2. *Attacking Procedure*

In this subsection, we apply the above basic idea and propose a novel differential fault analysis to recover the secret keys of LED-64 and LED-128. Our analysis is split into the following four successive steps:

(i) A ciphertext Y is derived when an arbitrary plaintext X is encrypted with a secret key K.

(ii) This step aims at recovering K_1 in the last round. The fault injection targets at the *l-2*-th round. As Fig. 2 shows, a fault may be induced on either AC^{l-2}, SC^{l-2} or SR^{l-2} whereas the approach is identical in either case. Note that any modification of one byte provokes the XOR-differences ΔMC^{l-2} on MC^{l-2}, ΔAC^{l-1} on AC^{l-1}, ΔSC^{l-1} on SC^{l-1}, ΔSR^{l-1} on SR^{l-1}, ΔMC^{l-1} on MC^{l-1}, ΔAC^{l} on AC^{l}, ΔSC^{l} on SC^{l}, ΔSR^{l} on SR^{l}, ΔMC^{l} on MC^{l}. These alter the original ciphertext Y into the faulty ciphertext Y^*. We can observe that

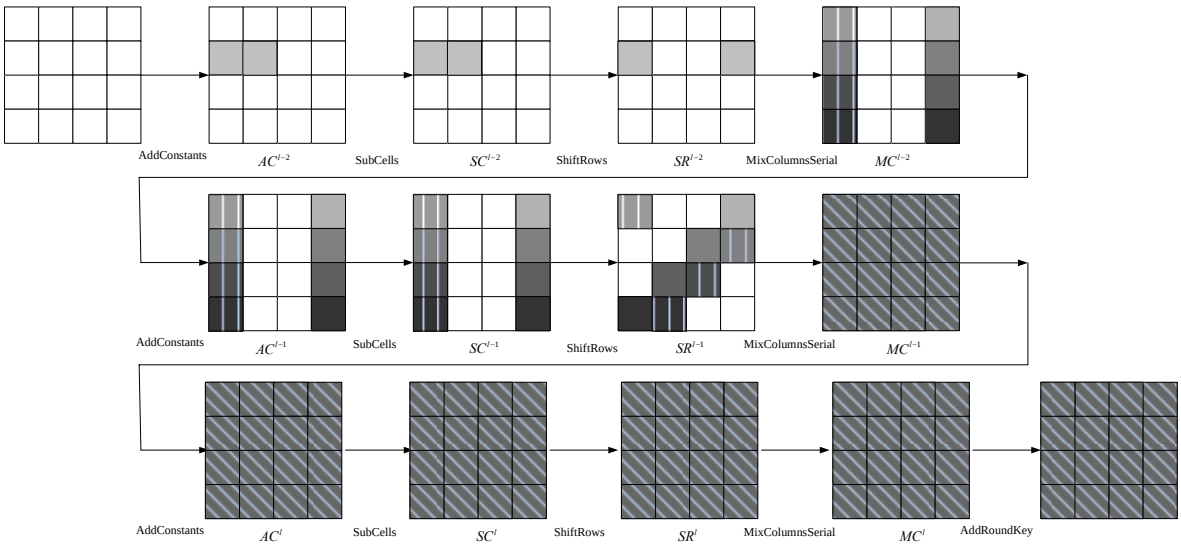

Fig. 2 One single-byte fault progagation path in the last three rounds of the LED cipher

$$\Delta Y = Y \oplus Y^*, \tag{4}$$

$$\Delta MC^l = (Y \oplus K_1) \oplus (Y^* \oplus K_1) = \Delta Y, \tag{5}$$

$$\begin{aligned} \Delta SC^l &= ShiftRows^{-1}(\Delta SR^l) \\ &= ShiftRows^{-1}(MixColumnsSerial^{-1}(\Delta MC^l)) \\ &= ShiftRows^{-1}(MixColumnsSerial^{-1}(\Delta Y)). \end{aligned} \tag{6}$$

The output difference of the *j*-th S-box in the *l*-th round could be represented by ΔSC_j^l, where $0 \le j \le 15$. The transformation between inputs difference and outputs difference of the *SubCells* transformation is defined in the *l*-th round as below:

$$\begin{aligned} IN(\Delta AC_j^l, \Delta SC_j^l) = \{ & AC_j^l \mid AC_j^l \in \{0,1\}^4, \\ & S(AC_j^l) \oplus S(AC_j^l \oplus \Delta AC_j^l) = \Delta SC_j^l,\ 0 \le j \le 15\}. \end{aligned} \tag{7}$$

The above equation, in conjunction with a pair of right and faulty ciphertexts, allow to infer a relation between ΔAC^l and ΔSC^l. It is helpful to restrict a list of possible candidates for the value of K_1. The *MixColumnsSerial* layer propagates one single byte fault to four-byte differences in the input of the SubCells transformation. If we do brute force search for the input of the *SubCells* transformation, the complexity to recover one subkey is up to 2^{64}. This kind of search is not really practical.

We propose an effective approach to select the input difference of the SubCells transformation, so the input of the SubCells transformation could be obtained with less complexity. We take the derivation of AC^l as an example. We observe that one single byte error can lead to two 4-byte differences independently after the computation of the diffusion layer in AC^{l-1}, and the differences could result in the 64-bit differences in ΔAC^l as Fig. 2 shows. This important property helps to do brute force search on ΔAC^l. We separate the one-byte fault into two 4-bit values. This approach simplifies the analysis and improve the attacking efficiency.

On the basis of one single-byte inducing fault, there are four types of the values of ΔAC^l in Fig. 3. In every type value, the set of ΔAC^l could be separated into two subsets of ΔAC^l which are easier to compute. On the basis of two subsets, the values of ΔAC^l shows the linear relationships between different columns.

The equations are derived as follows:

$$(\Delta Y_0, \Delta Y_7, \Delta Y_{10}, \Delta Y_{13}) = (S(AC_0^l) \oplus S(AC_0^l \oplus \Delta AC_0^l), S(AC_4^l) \oplus S(AC_4^l \oplus \Delta AC_4^l), \quad (8)$$
$$S(AC_8^l) \oplus S(AC_8^l \oplus \Delta AC_8^l), S(AC_{12}^l) \oplus S(AC_{12}^l \oplus \Delta AC_{12}^l)),$$

$$(\Delta Y_1, \Delta Y_4, \Delta Y_{11}, \Delta Y_{14}) = (S(AC_1^l) \oplus S(AC_1^l \oplus \Delta AC_1^l), S(AC_5^l) \oplus S(AC_5^l \oplus \Delta AC_5^l), \quad (9)$$
$$S(AC_9^l) \oplus S(AC_9^l \oplus \Delta AC_9^l), S(AC_{13}^l) \oplus S(AC_{13}^l \oplus \Delta AC_{13}^l)),$$

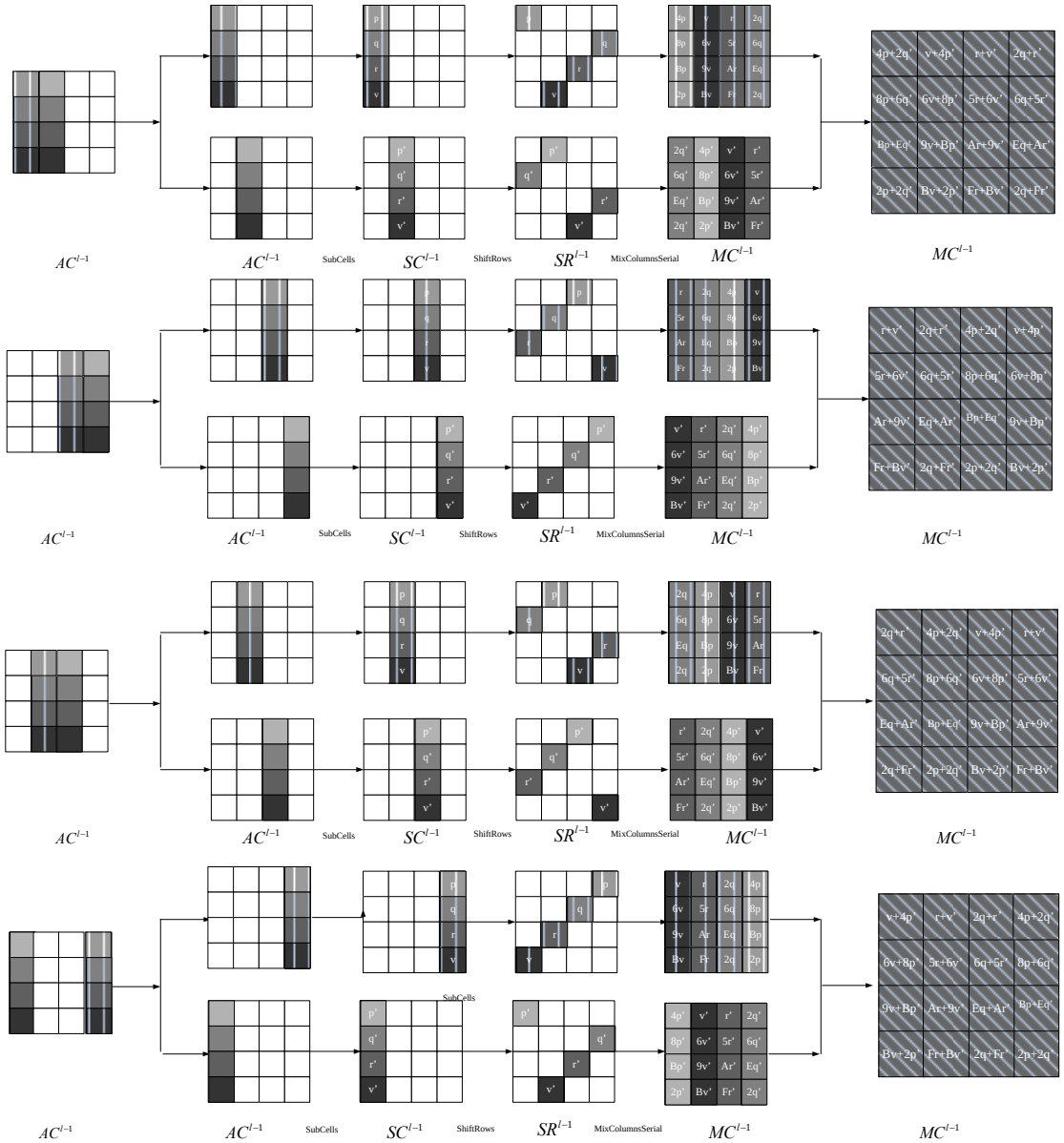

Fig. 3 The four relationships among the fault progagation values in the LED cipher

$$(\Delta Y_2, \Delta Y_5, \Delta Y_8, \Delta Y_{15}) = (S(AC_2^l) \oplus S(AC_2^l \oplus \Delta AC_2^l), S(AC_6^l) \oplus S(AC_6^l \oplus \Delta AC_6^l), \quad (10)$$
$$S(AC_{10}^l) \oplus S(AC_{10}^l \oplus \Delta AC_{10}^l), S(AC_{14}^l) \oplus S(AC_{14}^l \oplus \Delta AC_{14}^l)),$$

$$(\Delta Y_3, \Delta Y_6, \Delta Y_9, \Delta Y_{12}) = (S(AC_3^l) \oplus S(AC_3^l \oplus \Delta AC_3^l), S(AC_7^l) \oplus S(AC_7^l \oplus \Delta AC_7^l), \quad (11)$$
$$S(AC_{11}^l) \oplus S(AC_{11}^l \oplus \Delta AC_{11}^l), S(AC_{15}^l) \oplus S(AC_{15}^l \oplus \Delta AC_{15}^l)).$$

When the random fault influences the 1st, 2nd, 3rd or 4th columns of AC^{l-1}, the value of ΔAC^l could be represented by

$$\Delta AC^l = \begin{pmatrix} \Delta AC_0^l & \Delta AC_1^l & \Delta AC_2^l & \Delta AC_3^l \\ \Delta AC_4^l & \Delta AC_5^l & \Delta AC_6^l & \Delta AC_7^l \\ \Delta AC_8^l & \Delta AC_9^l & \Delta AC_{10}^l & \Delta AC_{11}^l \\ \Delta AC_{12}^l & \Delta AC_{13}^l & \Delta AC_{14}^l & \Delta AC_{15}^l \end{pmatrix} \in$$

$$\left\{ \begin{pmatrix} 4p+2q' & v+4p' & r+v' & 2q+r' \\ 8p+6q' & 6v+8p' & 5r+6v' & 6q+5r' \\ Bp+Eq' & 9v+Bp' & Ar+9v' & Eq+Ar' \\ 2p+2q' & Bv+2p' & Fr+Bv' & 2q+Fr' \end{pmatrix}, \right.$$

$$\begin{pmatrix} r+v' & 2q+v' & 4p+2q' & v+4p' \\ 5r+6v' & 6q+5r' & 8p+6q' & 6v+8p' \\ Ar+9v' & Eq+Ar' & Bp+Eq' & 9v+Bp' \\ Fr+Bv' & 2q+Fr' & 2p+2q' & Bv+2p' \end{pmatrix},$$

$$\begin{pmatrix} 2q+r' & 4p+2q' & v+4p' & r+v' \\ 6q+5r' & 8p+6q' & 6v+8p' & 5r+6v' \\ Eq+Ar' & Bp+Eq' & 9v+Bp' & Ar+9v' \\ 2q+Fr' & 2p+2q' & Bv+2p' & Fr+Bv' \end{pmatrix},$$

$$\begin{pmatrix} v+4p' & r+v' & 2q+r' & 4p+2q' \\ 6v+8p' & 5r+6v' & 6q+5r' & 8p+6q' \\ 9v+Bp' & Ar+9v' & Eq+Ar' & Bp+Eq' \\ Bv+2p' & Fr+Bv' & 2q+Fr' & 2p+2q' \end{pmatrix},$$

$$\left. p, p', q, q', r, r', v, v' \in \{0,1\}^4 \right\}. \quad (12)$$

Thus we could derive all possible values of AC^l. By inducing random faults repeating the above approach and until AC^l has only one value. All bytes of K_1 could be deduced as follows:

$$\begin{aligned} K_1 &= Y \oplus MC^l \\ &= Y \oplus MixColumnsSerial(SR^l) \\ &= Y \oplus MixColumnsSerial(ShiftRows(SubCells(AC^l))). \end{aligned} \quad (13)$$

(iii) After recovering the subkey K_1, we could perform the following procedure for K_2 of LED-128. We make advantage of the previous step (ii) to derive the output of the *l-4*-th rounds, and induce faults into the *l-6*-th round. After computing the input difference ΔAC^{l-4} and output difference ΔSC^{l-4} in the S-boxes transformation, we decrease the number of AC^{l-4} candidates by repeating the proposed method and the collected faulty ciphertexts, until the set of AC^{l-4} candidates has only one element. All bytes of AC^{l-4} could be deduced. Thus, all bytes of K_2 could be deduced as follows:

$$\begin{aligned} K_2 &= MC^{l-4} \oplus AR^{12} \\ &= MixColumnsSerial(ShiftRows(SubCells(AC^{l-4}))) \oplus \\ &\quad AddConstants(SubCells(ShiftRows(MixColumnsSerial(\\ &\quad AddConstants(SubCells(ShiftRows(MixColumnsSerial(\\ &\quad AddConstants(SubCells(ShiftRows(MixColumnsSerial(\\ &\quad AddConstants(SubCells(ShiftRows(MixColumnsSerial(\\ &\quad Y \oplus K_1)))))))))))))))). \end{aligned} \quad (14)$$

(iv) On the basis of the key schedule, $K=K_1$ in LED-64, and $K=K_1||K_2$ in LED-128.

5. Attacking Complexity

We summarize the attacking procedure to select subkey candidates for the 64-bit and 128-bit secret keys. The time complexity of brute-force search for one fault injection is

$$u = 2^{x \cdot s} \cdot (2^s \cdot \left\lceil \frac{n}{s} \right\rceil), \quad (15)$$

where n denotes the size of the *SubCells* transformation, s denotes the input size of one S-box, and x denotes the number of S-boxes in parallel.

In addition, an estimation of the number of faults necessary for the attack to be successful is vital. In the attacking procedure, the number of faulty ciphertexts to recover a subkey depends on the fault location and the fault model.

We take the derivation of K_1 as an example. On the property of the *SubCells* layer, if K_1 is a subkey candidate, then $K_1 \oplus \Delta MC^l$ may be another subkey candidate. In other words, if the input candidates set of S-boxes is not null, then the input AC^l may have several candidates. Thus, there are some candidates of MC^l. It indicates that K_1 may have some possible elements.

In the single-byte fault model, a random error could be induced at any round of the encryption. If the fault occurs in the last round, then only one single byte in the

input of the *SubCells* transformation will change, which could recover at most one byte of the last subkey by DFA. To recover the last subkey, it is necessary to induce many errors into different bytes.

If the fault is induced at an ideal location before the last round, then the inputs difference and outputs difference of the *SubCells* transformation in this round contain only one nonzero byte. However, the output difference of *ShiftRows* and *MixColumnsSerial* has multibytes owing to the diffusion of linear transformation. Thus, the input difference of *MixColumnsSerial* in the last round contains multibytes after the computation of the last several rounds. The above idea is applied in the attacking procedure to improve the efficiency of fault injection.

Since at least two errors can make one element in the intersection of K_1, we continue deriving intersection of subkey candidates sets until the intersection has only one element. Thus, at least two fault ciphertexts are required to derive multibytes of one subkey. The theoretical minimum number of faulty ciphertexts to recover one subkey is defined as

$$w = \begin{cases} 0 & if \quad \text{m=0}, \\ \left\lceil \dfrac{2n}{m} \right\rceil & if \quad 1 \le \text{m} \le \text{n}, \end{cases} \tag{16}$$

where n represents the size of the *SubCells* layer, and m represents the maximum number of bits in a subkey derived by two faulty ciphertexts. To derive the subkey, the value of m equals the number of bits in the nonzero output difference of the nonlinear transformation in this round. If $m = 0$, then there is no bits of a subkey derived and thus $w = 0$.

Thus, the overall attacking complexity to recover a secret key is

$$u \cdot w \cdot g = \begin{cases} 0 & if \quad m = 0, \\ 2^{(x+1)\cdot s+1} \cdot \left\lceil \dfrac{n^2 \cdot g}{s \cdot m} \right\rceil & if \quad 1 \le m \le n, \end{cases} \tag{17}$$

where g denotes the number of subkeys to recover a secret key, n denotes the size of the *SubCells* layer, s denotes the input size of one S-box, x denotes the number of S-boxes in parallel and m represents the maximum number of bits in a subkey derived by two faulty ciphertexts.

For a 64-bit secret key, the attacking complexity in theory is about

$$2^{25} (= 2^{(4+1)\cdot 4+1} \cdot \left\lceil \frac{64^2 \cdot 1}{4 \cdot 64} \right\rceil) \tag{18}$$

where g=1, n=64, x=4, s=4, m=64, and v=2. For a 128-bit secret key, the attacking complexity in theory is about

$$2^{26} (= 2^{(4+1)\cdot 4+1} \cdot \left\lceil \frac{64^2 \cdot 2}{4 \cdot 64} \right\rceil) \tag{19}$$

for g=2, n=64, x=4, s=4, m=64, and v=2.

6. Experimental Results

We implemented our attack on a PC using Visual C++ 8.0 Compiler on a 2.53 GHz celeron with 2GB memory. The fault induction was simulated by computer software. In this situation, we ran the attack algorithm to 1000 encryption unit with a random generated key.

Fig. 4 shows the number of subkey candidates in three intersections of subkey candidates to recover one subkey. We define accuracy, reliability and latency for evaluating the experimental results in detail.

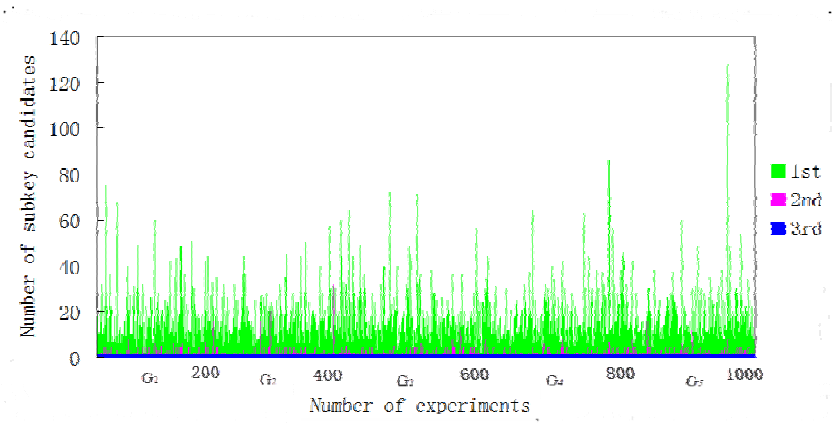

Fig. 4 (color) Three intersections of subkey candidates in the 1000 experiments

Accuracy is a measure that defines how close the number of subkey candidates are to the true number of subkey candidates. Basically, the closer the experimental number of subkey candidates is to the true number, the more accurate the experiment is. Thus, we consider the Root Mean-Square Error(RMSE) to measure the accuracy, where RMSE is given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{e=1}^{N}[h_{measured}(e) - h_{true}]}, \qquad (20)$$

where N is the number of experiments in a set and e is the index of the experiment, $h_{measured}$ is the number of subkey candidates, and h_{true} is the number of ture subkeys. As we know, there is only one true subkey. The closer the RMSE value is to 0, the more accurate the experiments are. We divide 1000 experiments as 5 groups in average, denoted as G_1, G_2, G_3, G_4 and G_5. The RMSE values for every intersections of subkey candidates are shown in Table 1, where N=200, $h_{true} = 1$ and $e \in \{1,\cdots,1000\}$. Thus, the 3rd intersection of subkey candidates is completely accurate, and we could derive the subkey in this intersection. That is, 4 fault ciphertexts are required to recover one subkey. Furthermore, the accuracy in every group for the same interaction is similar or equal.

Table 1. One subkey recovery on accuracy by RMSE

Groups	1st intersection	2nd intersection	3rd intersection
G_1	19.46	1.79	0
G_2	19.54	2.99	0
G_3	19.38	1.62	0
G_4	19.67	1.24	0
G_5	20.05	1.89	0

Table 2. One subkey recovery on reliability

Groups	1st intersection	2nd intersection	3rd intersection
G_1	0	65.0%	100%
G_2	0	64.0%	100%
G_3	0	66.0%	100%
G_4	0	67.5%	100%
G_5	0	65.5%	100%

Reliability is the ratio of successful experiments out of all experiments made. If the attacker could derive only one subkey, the experiment is successful. Referring to Table 2, it is observed that the ratio of successful experiments in the 1st, 2nd and 3rd intersections of subkey candidates are 0, 65.6% and 100%, respectively. That is, the reliability is 100% if the attacker induces 4 random faults to break a subkey. Furthermore, the reliability in every group for the same interaction is similar or equal.

Latency is the time from the first fault injection to the recovery of the subkey in our software simulation. It is measured in seconds. Fig. 5 shows that the latency of 1000 experiments. The time of 74.5% experiments is between 0.1s and 0.2s.

Thus, 4 faulty ciphertexts are required to recover one subkey. The proposed DFA method requires 4 faulty ciphertexts to recover the 64-bit secret key and 8 faulty ciphertexts to recover 128-bit secret keys.

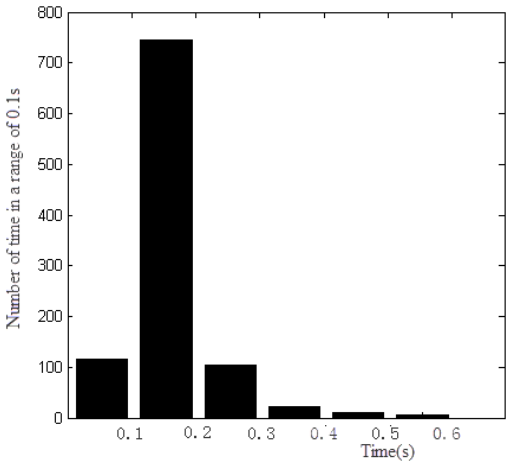

Fig. 5 One subkey recovery on latency

On the basis of the number of faulty ciphertexts in our simulated experiments, the attacking complexity in practice is

$$2^{26}(=2^{(4+1)\cdot 4}\cdot\left\lceil\frac{64^2\cdot 1}{4\cdot 64}\right\rceil\cdot 4) \qquad (21)$$

to recover one subkey. Thus, the attacking complexity is about

$$2^{26}(=2^{(4+1)\cdot 4}\cdot\left\lceil\frac{64^2\cdot 1}{4\cdot 64}\right\rceil\cdot 4) \qquad (22)$$

and

$$2^{27}(=2^{(4+1)\cdot 4}\cdot\left\lceil\frac{64^2\cdot 1}{4\cdot 64}\right\rceil\cdot 8) \qquad (23)$$

to break the LED-64 and LED-128 by the single byte differential fault analysis, respectively.

7. Conclusion

This paper examines single byte differential fault analysis on LED in software implementation. It shows that LED is vulnerable to the single byte differential

fault analysis. In the byte-oriented fault model, only 4 or 8 ciphertexts in average is required to obtain the 64-bit and 128-bit secret key of LED, respectively. Our work provides a new reference to fault analysis on other block ciphers.

In consequence, we are working on fault analysis on LED in hardware implementation. Furthermore, future analysis should be able to support more fault locations of LED, such as the key schedule.

Acknowledgements

The authors are grateful to the editors and the anonymous reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China under Grant No. 61003278 and No. 61073150, 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. J. Guo and T. Peyrin, A. Poschmann, et al. The LED block cipher, in *13th Int. W orkshop Cryptographic Hardware and Embedded Systems*, eds. B. Preneel and T. Takagi (Nara, Japan, 2011), pp. 326-341.
2. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems, in *17th Annual Int. Cryptology Conf.*, eds. S. Burton and Jr. Kaliski (California, USA, 1997) , pp. 513-525.
3. J. Blomer and J. P. Seifert. Fault based cryptanalysis of theadvanced encryption standard (AES), in *7th Int. Conf. Financial Cryptography*, eds. R. N. Wright (Guadeloupe, French West Indies, 2003), pp. 162-181.
4. C. N. Chen and S. M. Yen. Differential fault analysis on AES key schedule and some countermeasures. in *Proc. Australasian Conf. Information Security and Privacy*, eds. H. Wang, J. Pieprzyk and V. Varadharajan (Wollongong, Australia, 2004), pp. 118-129.
5. P. Dusart, G. Letourneux and O. Vivolo. Differential fault analysis on AES. in *1st Int. Conf . Applied Cryptography and Network Security* , eds. J. Zhou, M. Yung and Y. Han (Kunming, China, 2003), pp. 293-306.
6. C. Giraud. DFA on AES. in *4th Int. Conf. Advanced Encryption Standard*, eds. H. Dobbertin, V. Rijmen and A. Sowa (Bonn, Germany, 2004), pp. 27-41.
7. P. Gilles and J. J. Quisquater. A differential fault attack technique against SPN tsructures, with application to the AES and KHAZAD, in *5th Int. Workshop Cryptographic Hardware and Embedded Systems*, eds. C. D. Walter, C. L. Koc and C. Paar (Cologne, Germany, 2003), pp. 77-88.
8. M. Amir, T. M. S. Mohammad and S. Mahmoud. A generalized method of differential fault attack against AES cryptosystem. in *8th Int. Workshop Cryptographic Hardware and Embedded S ystems*, eds. L. Goubin and M. Matsui (Yokohama, Japan, 2006), pp. 91-100.
9. L. Hemme. A differential fault analysis against early rounds of (Triple-) DES, in 6th *Int. Workshop Cryptographic Hardware and Embedded Systems*, eds. M. Joye and J. J. Quisquater (Cambridge, MA, USA, 2004), pp. 254-267.
10. E. Biham, L. Granboulan and P. Q. Nguyon. Impossible fault analysis of RC4 and differential fault analysis of RC4, in *12th Int. Workshop Fast Software Encryption*, eds. H. Gilbert and H. Handschuh (Paris, France, 2005), pp. 359-367.
11. W. Li, D. Gu, J. Li, Z. Liu and Y. Liu. Differential fault analysis on Camellia. *J. Syst. Software* 83(2010) pp. 844-851.
12. W. Li, D. Gu and J. Li. Differential fault analysis on the ARIA algorithm, *Inform. Sciences* 178(19)(2008) 3727-3737.
13. L. Zhang and W. Wu, Differential fault analysis on SMS4. *Chinese J. Comput.* 29(9)(2008) 1596-1602.
14. R. Li, B. Sun, C. Li and J. You. Differential fault analysis on SMS4 using a single fault. *Inform. Process. Lett.* 111(4) (2011)156-163.
15. J. Li and D. Gu. Differential fault attack on PRESENT block cipher, in *Proc. Annual Conf. Chinese Association for Cryptologic Research*, eds. D. Pei and B. Yang (Guangzhou, China, 2009), pp. 3-13.