Algebraic Differential Fault Attacks on LED using a Single Fault Injection

Xinjie Zhao[a,*], Shize Guo[b], Fan Zhang[c], Tao Wang[a], Zhijie Shi[c], Keke Ji[a]

[a]*Department of Computer Engineering, Ordnance Engineering College, Shijiazhuang 050003, China*
[b]*The Institute of North Electronic Equipment, Beijing 100083,China*
[c]*Department of Computer Science and Engineering,University of Connecticut, Storrs 06269, USA*

Abstract

This paper proposes a new fault attack technique on the LED block cipher using a single fault injection by combining algebraic side-channel attack (ASCA) and differential fault attack (DFA). We name it as algebraic differential fault attack (ADFA). Firstly, a boolean equation set is constructed for LED using algebraic techniques. Then, the fault differences of the S-Box inputs in the last round of LED are deduced by DFA and represented using algebraic equations by the multiple deductions-based ASCA (MDASCA) technique proposed in COSADE 2012. Finally, the key is recovered by solving the equation set with the CryptoMiniSat solver. We show that, as to ADFA on LED under the single nibble-based fault model, the 64-bit key can be recovered within one minute on a common PC with a success rate of 79%, which is more efficient than previous work. We modify the CryptoMiniSat solver to count and output multiple solutions for the key, and conduct ADFA to calculate the reduced key search space for DFA. The key search space of LED is reduced to $2^6 \sim 2^{17}$, which is different from previous work. We also successfully extend ADFA on LED to other fault models using a single fault injection, such as byte based fault model and nibble based diagonal fault model, where traditional DFAs are difficult to work. The results show that ADFA is an efficient and generic fault analysis technique which significantly improves DFA.

Keywords: Algebraic cryptanalysis, Differential fault analysis, Single fault injection, CryptoMiniSat, LED.

1. Introduction

1.1. Background

The emerging pervasive computing demands have made low-end devices, such as smart cards, RFID tags, IC-printing applications become more and more popular. Such tiny computing devices are used in many applications and environments, leading to an ever increasing need for security in resource constrained environments. This has spurred the development of lightweight cryptography. Many ultra-lightweight block ciphers have been developed, such as mCrypton [21], DESXL [20], PRESENT [4], MIBS [12], KLEIN [8], PRINTcipher [16], Piccolo [26] and LED [9].

LED [9] is a hardware-optimized ultra-lightweight block cipher presented by J. Guo et al. in CHES 2011. It applies an AES-like design with the SPN structure, and only 966 GE are required

*Corresponding author. Tel.: +86 13643319969.
Email address: `zhaoxinjieem@163.com` (Xinjie Zhao)

for its implementation. The authors analyzed the security of LED under the main cryptographic analysis, and showed that LED is quite secure. However, the recent side-channel attacks (SCAs) bring a great number of threats on the implementation of cryptographic algorithms. Examples of practical SCAs are timing attacks [17], power attacks [18], electromagnetic radiation attacks [23], fault attacks [5] etc. It is necessary to investigate the security of LED against SCAs. LED has two variants with different key lengths: LED-64 and LED-128. In this paper, we mainly focused on the cryptanalysis of LED-64 against fault based active SCAs. In the rest of this paper, LED stands for LED-64 if not explicitly mentioned.

1.2. Previous work

Fault attacks retrieve the secret information by injecting a computational fault into the cryptosystem. Faults can be generated by changing the power supply voltage or the frequency of the external clock, varying the environmental temperature, and exposing the circuits of the device to intense lights or lasers during the computation [1]. The first idea was reported by Boneh et al. against the implementations of RSA-CRT in 1996 [5]. After that, Biham and Shamir proposed the differential fault analysis (DFA) attack by combining the fault attack with the differential cryptanalysis [3]. Since then, DFA has been applied to break public ciphers such as ECC [2], block ciphers such as AES [22] and CLEFIA [28], and stream ciphers such as RC4 [10] and Trivium [11] etc. Nowadays, DFA is considered as one of the most threatening attacks for the cryptographic systems.

DFA on LED was studied in [13] and [14] based on the main idea of DFA on AES in [29]. Both can break LED assuming a single nibble-based fault is injected into the 30-th round. However, different results on the reduced key search space are achieved. The work in [13] simply acclaimed the key search space can be reduced to 2^4 assuming the fault location is known. No experimental results were provided. The work in [14] showed that the key search space can be reduce to $2^{19} \sim 2^{25}$ with simulation experiments on an Opteron workstation having 48 GB RAM. Under the same fault model, the work in [19] studied algebraic fault attack (AFA) on LED, and on average 13.60 hours were required to reveal the key.

1.3. Our work

This paper makes a comprehensive study about DFAs on LED with algebraic techniques using a single fault injection. We initiate our study to improve the efficiency of fault attacks on LED under the same nibble-based fault model in [13, 14, 19]. A new technique called algebraic differential fault attack technique (ADFA) is proposed. ADFA combines DFA and algebraic techniques [6] (or algebraic side-channel attack: ASCA [24]) together. The fault differences of the S-Box inputs in the last round of LED are derived by DFA and represented with algebraic equations by the multiple deductions-based ASCA (MDASCA) technique proposed in COSADE 2012 [30]. The key is recovered by solving the equation set with a SAT solver. To accelerate the solving procedure, we show that it is important to build the algebraic equations for every reverse operation in LED instead of the original one. In our simulation experiments, the full master key can be recovered within one minute on a common PC for 79% cases, which is more efficient than previous work [14, 19].

Similar to [13] and [14], it is interesting to calculate the reduced key search space for ADFA on LED where an automatic and mathematic solver (the SAT-based CryptoMiniSat solver [27]) is used. However, the original solver will stop once a solution is found, and the solution may not be correct. To overcome this, we modify the CryptoMiniSat solver to count and output all the possible solutions. Under nibble-based fault model, the key search space of LED can be reduced to $2^6 \sim 2^{17}$, which is different from 2^4 in [13] and $2^{19} \sim 2^{25}$ in [14].

As ADFA is inherited with the generic feature of algebraic attacks, we are also interested in ADFAs on LED under other fault models where traditional DFAs are difficult to work. We consider byte-based fault model, diagonal-based fault model as in [29, 25] and only one fault injection is required to break LED.

The paper is organized as follows: Section 2 describes the design of LED. Section 3 presents our ADFA on LED. Section 4 describes the experimental results. Section 5 concludes the paper.

2. The LED Block Cipher

LED [9] is a lightweight block cipher with SPN structure, whose design has several AES-like features, for example, S-Boxes, ShiftRows, and MixColumns operations. The cipher state is conceptually arranged as a 4×4 matrix where each nibble represents an element from GF(2^4). For a 64-bit plaintext m, the 16 four-bit nibbles $m_0||m_1||\ldots||m_{15}$ are arranged as a 4×4 matrix. Likewise, the key is arranged as a 4×4 matrix and denoted as $K = k_0||k_1||\ldots||k_{15}$.

$$m = \begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{pmatrix}.$$

Like other block ciphers, LED is executed in a state updating trend. In the addRoundKey (AK), the plaintext P is Xored with K and then fed into a step operation. The value after repeating the step operation eight times will be Xored with K again, which is the ciphertext.

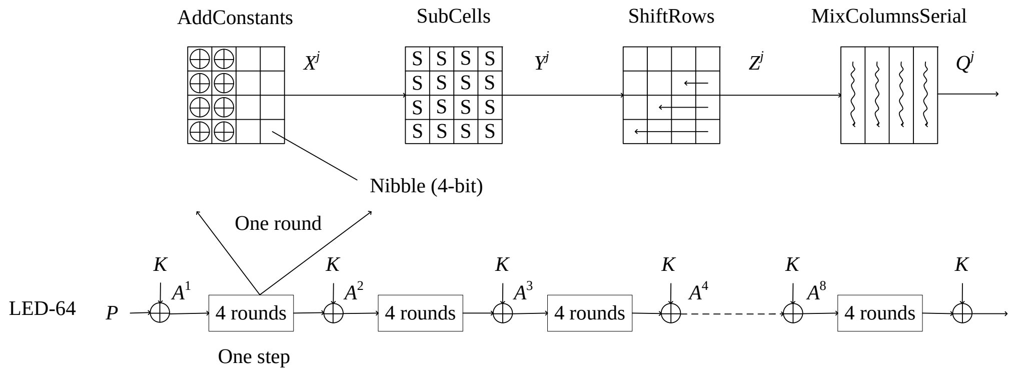


Figure 1: The specification of LED

The encryption procedure is illustrated in Fig. 1. Note that, there is no key schedule in LED. And one step operation consists of four round operations, each of which has four inner operations.

1. AddConstants (AC). The round constants are Xored with the state.
2. SubCells (SC). Each nibble in the state is updated by a lookup using the S-Box of PRESENT [4].
3. ShiftRows (SR). Row i of the state is rotated i distance to the left, for i = 0, 1, 2, 3.

4. MixColumnsSerial(MC). Each column of the state is updated by the multiplication of itself with the corresponding column of a fixed Matrix M.

$$M = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{pmatrix}.$$

3. Algebraic Differential Fault Analysis(ADFA) on LED

In this section, we describe the four phases of the Algebraic Differential Fault Analysis (ADFA) on LED. At first, we introduce the notations and fault model to be used in this paper.

3.1. Notations

We denote the output of the i-th AK as A^i, and the output of the j-th AC,SC,SR,MC function as X^j, Y^j, Z^j, Q^j, and the l-th nibble of A^i, X^j, Y^j, Z^j, Q^j as $A^i_l, X^j_l, Y^j_l, Z^j_l, Q^j_l$, where $1 \leq m \leq 9$, $1 \leq n \leq 32, 0 \leq l < 16$.

3.2. Fault Model

We assume that an attacker can inject only one random and nibble fault into Y^{30}, the output of SC in the 30-th round, as noted in [14, 19]. The value of the random fault is unknown. Fig. 2 depicts the propagation of injecting one nibble fault at the first nibble of Y^{30} in the last three rounds of LED. Every square in Fig. 2 depicts fault difference of one nibble of the intermediate state.

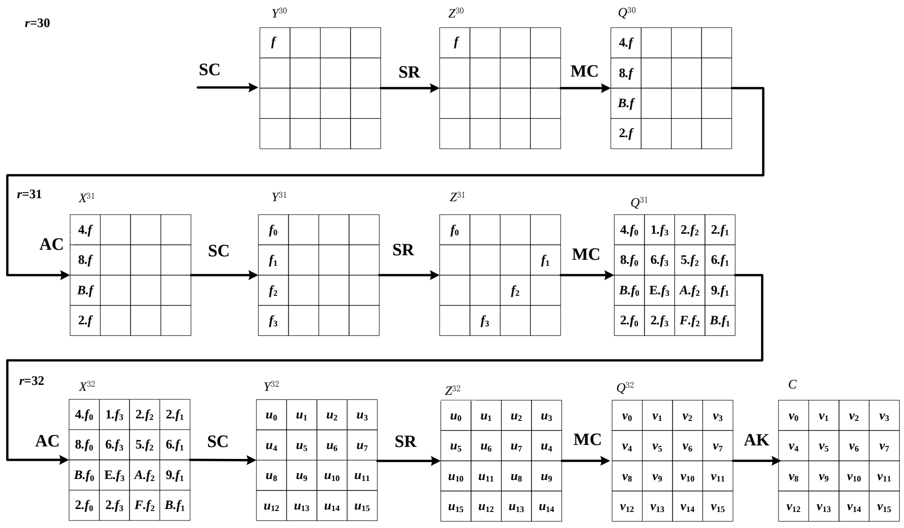


Figure 2: Fault model of ADFA on LED

We can see that: (1) AC and AK affect neither the values nor the locations of the fault difference. (2) SC only changes the values of the fault difference, (3) SR only changes the locations of the fault difference, (4) MC changes both the the values and locations.

Among the four operations, only the fault differences of the outputs of SC are associated with both the state and its inputs, which can be used to recover the value of the state and then reveal the key. As the output fault difference of SC in the 32-th round, $\triangle Y^{32}$ can be computed directly from the ciphertext difference. The key issue of fault attacks on LED is to derive the input fault difference $\triangle X^{32}$ from $\triangle Y^{32}$. We will present the details in Section 3.4.

3.3. Building the equation set of LED

The goal of this phase is to transform LED into a big system of low degree boolean equations. In this system, the key bits appear as variables and solving the system is equivalent to recovering them. Two strategies can be considered. The first is to directly build the equations for every forward operation, and the second is to build the algebraic equations for every reverse operation. Differences of solving time for these two strategies will be discussed in Section 4.1.

When building the equation set of LED, how to represent SC and MC (and their reverse functions SC^{-1} and MC^{-1}) is the most challenging among all operations. We describe them in the following.

(1) Represent SC and SC^{-1}.

SC can be described by 16 table lookups to a S-Box. In this paper, we exploit the techniques in [15] to derive every S-Box output bit with high-degree equations from the four S-Box input bits. Suppose (x_0, x_1, x_2, x_3) and (y_0, y_1, y_2, y_3) denote the input and output of the 4-bit S-Box. Then, the S-Box lookup can be denoted as

$$\begin{aligned}
y_0 &= 1 + x_0 + x_2 + x_3 + x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3 \\
y_1 &= 1 + x_0 + x_1 + x_0x_2 + x_0x_3 + x_2x_3 + x_0x_1x_3 + x_0x_2x_3 \\
y_2 &= x_0 + x_2 + x_0x_1 + x_0x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3 \\
y_3 &= x_0 + x_1 + x_3 + x_1x_2
\end{aligned} \tag{1}$$

The reverse of S-Box lookup can be denoted as

$$\begin{aligned}
x_0 &= y_0 + y_1 + y_2 + y_3 + y_2y_2 + y_0y_1y_3 + y_1y_2y_3 \\
x_1 &= 1 + y_0 + y_1y_2 + y_0y_2 + y_1y_3 + y_2y_3 + y_0y_3 + y_1y_2y_3 + y_0y_1y_3 + y_0y_2y_3 \\
x_2 &= y_0 + y_2 + y_3 + y_0y_1 + y_0y_2 + y_1y_3 + y_1y_2y_3 + y_0y_1y_3 + y_0y_2y_3 \\
x_3 &= 1 + y_1 + y_3 + y_0y_2
\end{aligned} \tag{2}$$

(2) Represent MC and MC^{-1}.

Firstly, we present M^{-1}, the revise matrix of M.

$$M^{-1} = \begin{pmatrix} C & C & D & 4 \\ 3 & 8 & 4 & 5 \\ 7 & 6 & 2 & E \\ D & 9 & 9 & D \end{pmatrix}.$$

When representing MC and MC^{-1}, the comparatively difficult part is to represent the element multiplication in M and M^{-1}. Suppose (x_0, x_1, x_2, x_3) and (y_0, y_1, y_2, y_3) denote the input and output of the element multiplication in M, M^{-1}. Their relations can be represented in Table 1.

Table 1: Represent element multiplication in M, M^{-1}

Matrix element	y_0	y_1	y_2	y_3
1	x_0	x_1	x_2	x_3
2	x_1	x_2	x_0+x_3	x_0
3	x_0+x_1	x_1+x_2	$x_0+x_2+x_3$	x_0+x_3
4	x_2	x_0+x_3	x_0+x_1	x_1
5	x_0+x_2	$x_0+x_1+x_3$	$x_0+x_1+x_2$	x_1+x_3
6	x_1+x_2	$x_0+x_2+x_3$	x_1+x_3	x_0+x_1
7	$x_0+x_1+x_2$	$x_0+x_1+x_2+x_3$	$x_1+x_2+x_3$	$x_0+x_1+x_3$
8	x_0+x_3	x_0+x_1	x_1+x_2	x_2
9	x_3	x_0	x_1	x_2+x_3
A	$x_0+x_1+x_3$	$x_0+x_1+x_2$	$x_0+x_1+x_2+x_3$	x_0+x_2
B	x_1+x_3	x_0+x_2	$x_0+x_1+x_3$	$x_0+x_2+x_3$
C	$x_0+x_2+x_3$	x_1+x_3	x_0+x_2	x_1+x_2
D	x_2+x_3	x_3	x_0	$x_1+x_2+x_3$
E	$x_0+x_1+x_2+x_3$	$x_1+x_2+x_3$	x_2+x_3	$x_0+x_1+x_2$
F	$x_1+x_2+x_3$	x_2+x_3	x_3	$x_0+x_1+x_2+x_3$

3.4. Deducing the fault differences with DFA

Let $\triangle$ denote the function of computing the fault difference of the interemidate state. As noted in Section 3.2, deducing $\triangle X^{32}$ from $\triangle Y^{32}$ is the crucial part for the key recovery of DFA on LED. From Fig. 2, $\triangle Y^{32} = u_0||u_1||\ldots||u_{15}$ can be easily calculated from the ciphertext difference $\triangle C$.

$$\triangle Y^{32} = SR^{-1}(MC^{-1}(\triangle C)) \tag{3}$$

From Fig. 2, we need to deduce the value of four nibbles f_0, f_1, f_2, f_3 in order to calculate $\triangle X^{32}$. We use u_0, u_4, u_8, u_{12} to deduce f_0, u_1, u_5, u_9, u_{13} to deduce f_3, u_2, u_6, u_{10}, u_{14} to deduce f_2, u_3, u_7, u_{11}, u_{15} to deduce f_1. Next, we take deducing f_0 as an example to describe our technique.

Step 1: For each possible candidate of f_0 ($1 \le f_0 \le 15$) and S-Box element a ($0 \le a \le 15$), compute the differential S-Boxes $S_0[f_0-1], S_4[f_0-1], S_8[f_0-1], S_{12}[f_0-1]$ corresponding to $4 \cdot f_0, 8 \cdot f_0, B \cdot f_0, 2 \cdot f_0$.

$$\begin{aligned} S_0[f_0-1][a] &= S[4 \cdot a] \oplus S[4 \cdot (a \oplus f_0)] \\ S_4[f_0-1][a] &= S[8 \cdot a] \oplus S[8 \cdot (a \oplus f_0)] \\ S_8[f_0-1][a] &= S[B \cdot a] \oplus S[B \cdot (a \oplus f_0)] \\ S_{12}[f_0-1][a] &= S[2 \cdot a] \oplus S[2 \cdot (a \oplus f_0)] \end{aligned} \tag{4}$$

Step 2: For each possible f_0 candidate, if u_0, u_4, u_8, u_{12} are all in the joint set composed by elements of $S_0[f_0-1], S_4[f_0-1], S_8[f_0-1], S_{12}[f_0-1]$, then this candidate is kept for f_0. Else, just discard it.

Applying the two steps above, 1-4 candidates of f_0 can be deduced and used for further key recovery. f_1, f_2, f_3 can be deduced using the similar steps. Details of statistics on the number of candidates for f_0, f_1, f_2, f_3 will be described in Section 4.1.

3.5. Representing the fault differences with MDASCA

Representing the fault difference $\triangle X^{32}$ with algebraic equations is easy if $\triangle X^{32}$ is single and correct. However, this does not hold in fault attacks on LED. Multiple deductions on the value of

$\triangle X_i^{32}$ ($0 \le i \le 15$) can be deduced in practice. How to represent these multiple deductions is very important in ADFA on LED.

In COSADE 2012, Zhao et al. proposed the multiple deductions-based algebraic side-channel attack (MDASCA) [30] to represent multiple deductions of the intermediate states in the cryptosystems, which can also be used to tolerate errors and exploit new leakage model (e.g., cache leakage models). In this section, we adopt MDASCA to represent the multiple values of $\triangle X_i^{32}$ and exploit fault models.

Let $d = d^0, d^1, d^2, d^3$ denote the correct deduction of $\triangle X_i^{32}$. Let $D = d_1, d_2, \ldots, d_n$ denote the possible deduction set on d, d_i be the i-th element in D, the size of D is n. Then, algebraic equations about d and d_i can be built as followings.

Step 1: Representing d_i. Let d_i^j be the j-th bit of d_i, then $4n$ new one-bit variables are introduced to represent d_i.

Step 2: Representing the relations on d and d_i.

A one-bit variable c_i is introduced to represent whether d_i is equal to d or not. Another one-bit variable e_i^j is also introduced to represent whether ${d_i}^j$ is equal to d^j. Then c_i can be represented with Eq.(5), where $\neg$ denotes the NOT operation.

$$e_i^j = \neg(d_i^j \oplus d^j), \quad c_i = \prod_{j=1}^{b} e_i^j \tag{5}$$

As only one d_i is equal to d (c_i is 1 then), which can be represented as:

$$c_1 \vee c_2 \vee \ldots \vee c_{s_p} = 1, \quad \neg c_i \vee \neg c_j = 1, \quad 1 \le i < j \le s_p \tag{6}$$

As shown in this section, the algebraic equations for new constraints are quite simple. They can be easily fed into a solver to recover the key.

3.6. Solving for the master key

Many automatic tools can be used for this phase, such as Gröbner basis-based [7], or SAT-based solver [27]. In this paper, we use a SAT-based solver, CryptoMiniSat 2.9.4 [27], running on a quad-core Intel I5 2400 clocked at 3.10 GHz and 32-bit Windows XP operating system.

4. Experimental results

To verify the effects of ADFA on LED, we conduct many experiments and report the results in this section. How to inject a real fault was widely studied in many previous work [1, 29] and is not the major concern of this paper. We simulate the fault injection by the computer software and calculate the solving time under the setup in Section 3.6. One full ADFA on LED is denoted as an *instance*.

4.1. ADFA on LED under nibble-based fault model

For each instance, we first generate a random plaintext P, secret key K and get the correct ciphertext C. Then, we inject one random nibble fault at Y^{30} when encrypting P with the same K and get the faulty ciphertext C^*. Finally, we use P, C and C^* to deduce K.

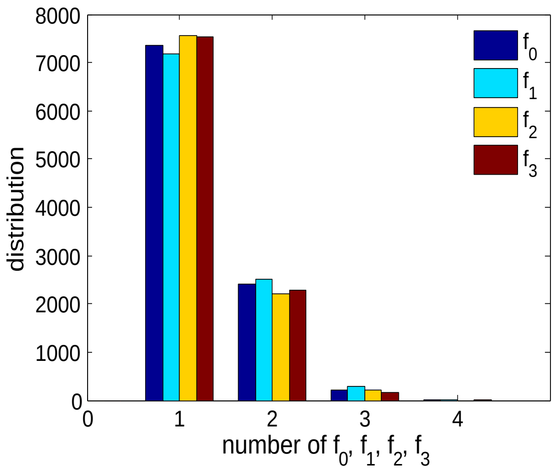
To calculate the distributions of the number of f_0, f_1, f_2, f_3 in ADFA on LED, we generate random keys, plaintexts and faults for 10000 times and use the proposed DFA technique to deduce

Table 2: Comparisons of ADFA on LED with previous work

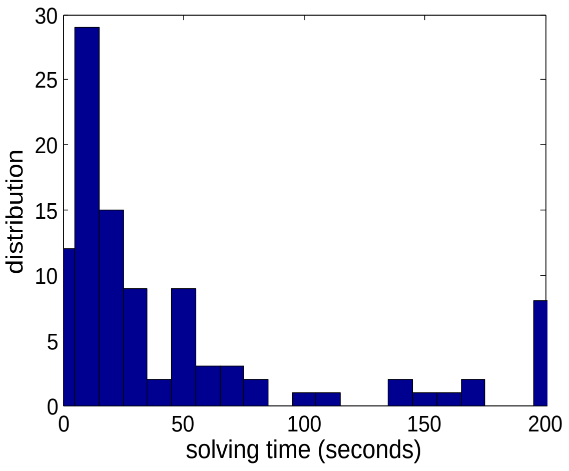
Attack	Technique	Time complexity	Attack setup	Key search space
[13]	DFA	–	–	2^4
[14]	DFA	45s in preprocessing phase	Opteron workstation having 48 GB RAM	$2^{19} \sim 2^{25}$
[19]	AFA	13.6 hours	eight-core Xeon processor	1
This paper	**ADFA**	**1-3 minutes**	**quad-core Intel processor**	**1**

f_0, f_1, f_2, f_3. The distributions on the number of f_0, f_1, f_2, f_3 are shown in Fig. 3(a). We can see that multiple candidates (1-4) of f_0, f_1, f_2, f_3 can be deduced in practice, and the distributions on the number of f_0, f_1, f_2, f_3 are slightly different (affected by the MC function).

In ADFA on LED, we build the algebraic equations of a full round LED for the correct encryption (both P and C are fed into the equations) and the last 3 rounds for the faulty encryption (only C^* is fed into the equations). At the beginning, we adopt the first strategy in Section 3.3 to build the algebraic equations and run the instances for about 10 times. On average 15 hours are required to recover the correct key, which is also consistent with the results in [19]. Then, we adopt the second strategy to build the algebraic equations for the reverse operations in LED and run the instances for about 100 times. The calculated distribution of time complexity of the attack is shown in Fig. 3(b).



(a) Distribution on the number of f_0, f_1, f_2, f_3



(b) Distribution of the time complexity

Figure 3: Results of ADFA on LED

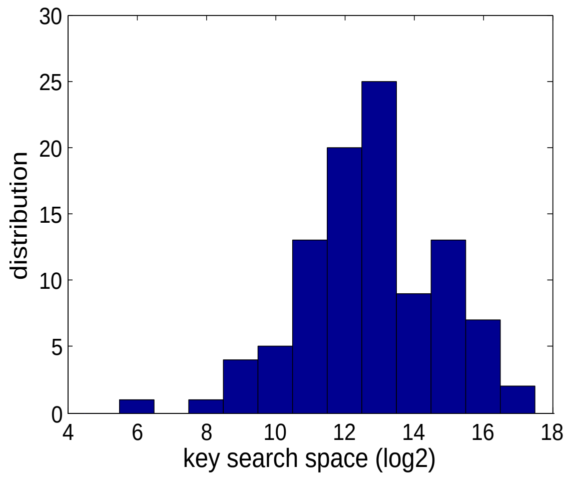
From Fig. 3(b), we can see that (1) the equation solving time of ADFA on LED seems to follow an exponential distribution (as noted in [24, 30]) and most of the instances can be solved within one minute, (2) the correct key can be solved within one minute with a success rate of 79%, and three minutes with that of 92%. In fact, if we set the threshold as 10 minutes, the success rate is 100%. The comparisons of ADFA on LED with previous work are shown in Table 2.

As [13] provides no results and details of DFA on LED, we just compare our result with [14] and [19]. Compared with [14], the CryptoMiniSat solver in ADFA can automatically solve for key and output the whole key at a time, and no extra search on the master key is required, as in [19]. Meanwhile, the attack setup of our ADFA is less costive and the time of the full attack is also comparable. Compared with [19], the time required in ADFA is much less. The experimental results also indicate that to use the second strategy can significantly improve the efficiency of ADFA on LED.

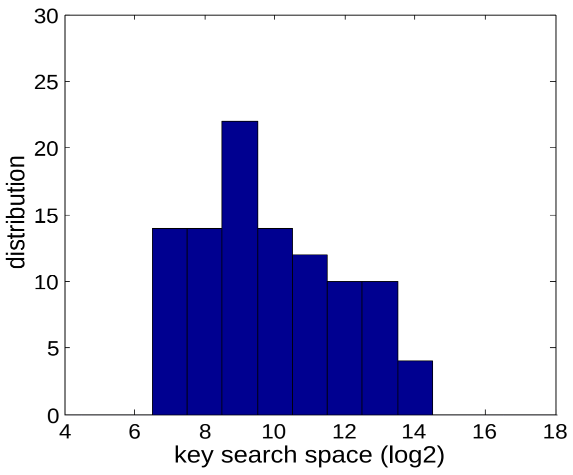
4.2. Evaluating the key research space of DFA on LED with ADFA

To evaluate the key search space to be reduced in DFA for a given fault model is very important to test the resistances of ciphers against fault attacks. We are interested in finding an automatic way to do this. Compared with the manual calculation, the solver approach in ADFA is more reliable and robust.

In DFA, the key search space is only calculated by analyzing correct and faulty ciphertexts. To calculate the reduced key research space of DFA on LED with ADFA, we only build the algebraic equations for the last 3 rounds of LED. The value of C and C^* is also fed into the solver. Under this attack scenario, the original CryptoMiniSat sovler always outputs a satisfiable but wrong solution and stops, which foils the attack.



(a) Nibble-based fault model



(b) Byte-based fault model

Figure 4: Reduced key search space of ADFA on LED

Thanks to the help from Mate Soos and Martin Maurer, the designers of CryptoMiniSat. We modify the source code of the solver, and enable it to support counting and outputting all the possible key solutions in ADFA on LED automatically. We run the instances for about 100 times and calculate the distribution on the number of the possible solutions for the master key, as shown in Fig. 4(a). The key search space of LED is reduced to $2^6 \sim 2^{17}$, which is less than $2^{19} \sim 2^{25}$ in [14] and more than the theoretical value 2^4 in [13].

4.3. Extending ADFA on LED to other fault models

Inherited with the advantages of algebraic attacks, ADFA is much more generic than the traditional DFA. We are also interested in extending ADFA on LED to other fault models where traditional DFAs are difficult to work. More specifically, two fault models are considered.

(1) Byte-based fault model. Under this model, the fault propagation pattern is much more complicated than nibble-based fault model. It is difficult to derive the $\triangle X^{32}$ from $\triangle Y^{32}$ with the

DFA method in [13] and [14]. In out attack, we just represent the ciphertext difference and all the zero fault differences in the fault propagation procedure. We run the instances for about 100 times. The solver can output the 64-bit master key with on average one hour. We also evaluate the reduced key research space of DFA on LED under byte-based fault model, as shown in Fig. 4(b). It is interesting to find out that the key search space can be reduced to $2^6 \sim 2^{14}$, which is even less than that in nibble-based fault model.

(2) Diagonal-based fault model. Diagonal-based fault model is a classic fault model in DFA on AES[25], in which the fault is induced into the diagonals (one to four matrix elements) of the state matrix, as proposed in [25]. Different from [25], we adopted the nibble-based fault model instead of the byte-based fault model.

Let $S = s_1||s_2||\dots||s_{15}$ denote the state matrix in LED. A diagonal is a set of four elements of S, where the i-th diagonal D_i is defined as: $D_0 = s_0, s_5, s_{10}, s_{15}$, $D_1 = s_1, s_6, s_{11}, s_{12}$, $D_2 = s_2, s_7, s_8, s_{13}$, $D_4 = s_3, s_4, s_9, s_{14}$. Under the assumptions of injecting one diagonal fault (one to four nibbles become faulty), we run the ADFA instances for about 100 times and on average one hour is enough to break LED.

5. Conclusions and Future Work

This paper presents a new fault analysis technique, named as algebraic differential fault attack technique (ADFA) and applies it to LED block cipher. We show that LED can be broken with only one fault injection with less cost and time complexity than previous work [14, 19]. In addition, we modified CryptoMiniSat solver to support multiple solutions and apply it in ADFA to evaluate the reduced key search space in DFA. Different results have been achieved compared to previous work.

Meanwhile, we also successfully extend ADFA on LED to other complicated fault models. Note that ADFA can also be extended to improve DFA on other lightweight block ciphers. Using a single nibble fault injection, we have also successfully conducted ADFA on MIBS [12], Klein [8] and Piccolo [26]. More details will be reported in upcoming papers.

The experimental results of this paper show that ADFA is both efficient and generic. How to analytically estimate the reduced key search space of DFA on LED under different fault models in this paper, how to apply ADFA on AES and how to use CryptoMiniSat solver to evaluate the reduced key search space under different leakage models in ASCA [24, 30] are interesting problems to explore in the future. Meanwhile, we are also planning to implement LED in hardware and conduct the attacks with physical devices using different fault models.

Acknowledgments.

The authors would like to thank Mate Soos and Martin Maurer for their helpful comments and efforts in modifying the CryptoMiniSat solver. This work was supported in part by the National Natural Science Foundation of China under the grants 60772082 and 61173191, and US National Science Foundation under the grant CNS-0644188.

References

[1] H. Bar-El, H. Choukri,D. Naccache, M. Tunstall, C. Whelan. The Sorcerers Apprentice Guide to Fault Attacks. In IEEE 94, pp. 370-382, 2006.

[2] I. Biehl, B. Meyer, V. Muller. Differential fault analysis on elliptic curve cryptosystems. In CRYPTO 2000. LNCS, vol. 1880, pp. 131-146, 2000.

[3] E. Biham, A. Shamir. Differential Fault Analysis of Secret Key Cryptosystem. In CRYPTO 1997, LNCS, vol. 1294, pp. 513-525, 1997.

[4] A. Bogdanov, L.R. Knudsen, G. Leander, et al. PRESENT: An Ultra-Lightweight Block Cipher. In CHES 2007. LNCS, vol. 4727, pp. 450-466, 2007.

[5] D.Boneh, R.A.DeMillo, R.J.Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In EUROCRYPT 1997, LNCS, vol. 1233, pp. 37-51, 1997.

[6] N. Courtois, J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In ASIACRYPT 2002, LNCS, vol. 2501, pp. 267-287, 2002.

[7] J.-C. Faugère, Gröbner Bases. Applications in Cryptology. in proceedings of FSE 2007 Invited Talk, available at: http://fse2007.uni.lu/slides/faugere.pdf.

[8] Z. Gong, S.I. Nikova, Y.W. Law. KLEIN: A New Family of Lightweight Block Ciphers. Technical Report TR-CTIT-10-33, Centre for Telematics and Information Technology, University of Twente, Enschede. ISSN 1381-3625.

[9] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw. The LED Block Cipher. In CHES 2011, LNCS, vol. 6917, pp. 326-341, 2011.

[10] J.J. Hoch, A.Shamir. Fault analysis of stream ciphers. In CHES 2004, LNCS, vol. 3156, pp. 240-253, 2004.

[11] M. Hojsik and B. Rudolf., B.: Differential fault analysis of Trivium. In FSE 2008, LNCS, vol. 5086, pp. 58-172, 2008.

[12] M. Izadi, B. Sadeghiyan, S.S. Sadeghian, et al. MIBS: A New Lightweight Block Cipher. In CANS 2009, LNCS, vol. 5888, pp. 334-348, 2009.

[13] K. Jeong and C. Lee. Differential Fault Analysis on Block Cipher LED-64. Future Information Technology, Application, and Service, LNEE, vol. 164, pp.747-755, 2012.

[14] P. Jovanovic, M. Kreuzer, and I. Polian. A Fault Attack on the LED Block Cipher. COSADE 2012, LNCS, vol. 7275, pp. 120-134, 2012.

[15] L.R. Knudsen, C.V. Miolane. Counting equations in algebraic attacks on block ciphers. International Journal of Information Security , vol. 9, No. 2, pp. 127-135, 2010.

[16] L. Knudsen, G. Leander, A. Poschmann, et al. PRINTcipher: A Block Cipher for IC-Printing. In CHES 2010, LNCS, vol. 6225, pp. 16-32, 2010.

[17] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In CRYPTO 1996, LNCS, vol. 1109, pp. 104-113, 1996.

[18] P. Kocher, J.Jaffe, B. Jun. Differential power analysis. In CRYPTO 1999, LNCS, vol. 1666, pp. 388-397, 1999.

[19] M. Kreuzer. Algebraic Fault Attacks Webinar. In Symbolic Computation and Post-Quantum Cryptography 2012, Apr 19, 2012.

[20] G. Leander, C. Paar, A. Poschmann, et al. New Lightweight DES Variants. In FSE 2007, LNCS, vol. 4593, pp. 196-210, 2007.

[21] C. Lim, T. Korkishko. mCrypton - A Lightweight Block Cipher for Security of Low-cost RFID Tags and Sensors. In WISA 2005, LNCS, vol. 3786, pp. 243-258, 2006.

[22] G. Piret, J.J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad. In CHES 2003, LNCS, vol. 2779, pp. 77-88, 2003.

[23] J. J. Quisquater, D. Samyde. A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions: the SEMA and DEMA methods, Eurocrypt rump session, 2000.

[24] M. Renauld, F.-X. Standaert. Algebraic Side-Channel Attacks. In INSCRYPT 2009, LNCS, vol. 6151, pp. 393-410, 2009.

[25] D. Saha, D. Mukhopadhyay, and D. RoyChowdhury, A Diagonal Fault Attack on the Advanced Encryption Standard. Cryptology ePrint Archive. Available: http://eprint.iacr.org/2009/581.pdf, 2009.

[26] K. Shibutani, T. Isobe, H. Hiwatari, et al. Piccolo: An Ultra-Lightweight Blockcipher. In CHES 2011, LNCS, vol. 6917, pp. 342-357, 2011.

[27] M. Soos, K. Nohl, and C. Castelluccia. Extending SAT Solvers to Cryptographic Problems. In SAT 2009, LNCS, vol. 5584, pp. 244-257, 2009.

[28] J. Takahashi and T. Fukunaga. Improved Differential Fault Analysis on CLEFIA. In FDTC 2008, pp. 25-34, 2008.

[29] M. Tunstall, D.Mukhopadhyay, S. Ali. Differential fault analysis of the advanced encryption standard using a single fault. In WISTP 2011, LNCS, vol. 6633, pp. 224-233, 2011.

[30] X. J. Zhao, S. Z. Guo, F. Zhang , et al. MDASCA: An Enhanced Algebraic Side-Channel Attack for Error Tolerance and New Leakage Model Exploitation. In COSADE 2012, LNCS, vol. 7275, pp. 231-248, 2012.