An Improved Differential Fault Attack on Camellia*

ZHAO Xin-jie, WANG Tao

(Department of Computer Engineering, Ordnance Engineering College, Shijiazhuang 050003, China)

zhaoxinjieem@163.com

Abstract: The S-box lookup is one of the most important operations in cipher algorithm design, and also is the most effective part to prevent traditional linear and differential attacks, however, when the physical implementation of the algorithm is considered, it becomes the weakest part of cryptosystems. This paper studies an active fault based implementation attack on block ciphers with S-box. Firstly, it proposes the basic DFA model and then presents two DFA models for Feistel and SPN structure block ciphers. Secondly, based on the Feistel DFA model, it presents several improved attacks on Camellia encryption and proposes new attacks on Camellia key schedule. By injecting one byte random fault into the r-1th round left register or the the r-1th round key, after solving 8 equations to recover 5 or 6 propagated differential fault of the rth round left register, 5 or 6 bytes of the rth equivalent subkey can be recovered at one time. Simulation experiments demonstrate that about 16 faulty ciphertexts are enough to obtain Camellia-128 key, and about 32, 24 ciphertexts are required to obtain both Camellia-192/256 key with and without FL/FL^{-1} layer respectively. Compared with the previous study by ZHOU Yongbin et. al. by injecting one byte fault into the rth round left register to recover 1 equivalent subkey byte and obtaining Camellia-128 and Camellia-192/256 with 64 and 96 faulty ciphertexts respectively, our attacks not only extend the fault location, but also improve the fault injection efficiency and decrease the faulty ciphertexts number, besides, our DFA model on Camellia encryption can be easily extended to DFA on Camellia key schedule case, while ZHOU's can not. The attack model proposed in this paper can be adapted into most of the block ciphers with S-boxes. Finally, the contradictions between traditional cryptography and implementation attacks are analyzed, the state of the art and future directions of the DFA on Block ciphers with S-boxes are discussed.

Key words: Implementation attack; Differential fault analysis; S-box lookup; Feistel structure; SPN structure; Camellia; Block cipher Key schedule

1 Introduction

1.1 Related Works

Cryptology is the science that studies how to hide confidential information. Cryptology comprises of two complementary fields. Cryptography is the study and the practice of hiding information through designing ciphers and protocols, while cryptanalysis is the study of methods to obtain knowledge from hidden information. Traditionally speaking, ciphers are designed so that by observing only the input and output of the algorithm it is computationally infeasible to break the cipher, or equivalently determine the secret key used in encryption and decryption. Thus, the algorithm itself does not leak enough useful information during its operation to compromise security. With the development of Cryptography and Cryptanalysis, both the key length and algorithm complexity of cipher have been greatly improved, so it's very difficult to compromise it through mathematically analysis. However, when a physical implementation of the algorithm is considered, additional information like timing of the circuit implementing the algorithm, behavior as a result of internal faults[2], power consumption[3], electromagnetic emanation[4], and acoustic effects[5] can provide enough information to compromise the security of the system. Attacks based on the use of this implementation specific information are known as Implementation Attacks (IMA), and they are quite effective, most of the ciphers are facing serious challenges. New directions for Cryptology are emerged to Implementation Cryptography and Implementation Cryptanalysis. The attacks present in this paper use Fault Cryptanalysis of block cipher.

The idea of fault attack was first suggested in 1997 by Boneh, DeMillo and Lipton[2], which makes use of the faults during the execution of a cryptographic algorithm. Under the idea, the attack was successfully exploited to break an RSA-CRT with both a correct and a faulty signature of the same message. Shortly after, Biham and Shamir proposed an attack on secret key cryptosystems called Differential Fault Analysis (DFA)[6], which combined the ideas of fault attack and differential attack. DFA attacks are based on deriving information about the secret key by examining the differences between ciphers resulting from correct operations and faulty operations. Since the presentation of DFA, many research papers have been published on using this cryptanalysis technique to successfully attack various cryptosystems, including ECC[7], 3DES[8], AES[9][10][11][12][13][14][15], Camellia[16], ARIA[17], CLEFIA[18][19],SMS4[20][21][22],

* Supported by the National Natural Science Foundation of China (Grant No. 60772082) and the Natural Science Foundation of Hebei Province. China (Grant No. 08M010). ZHAO Xin-jie, male, born in 1986, Ph.D.candidate, his main research interests include: implementation attack analysis on symmetric ciphers, formal methods analysis; WANG Tao, male, born in 1964, Ph.D., professor, Ph.D. supervisor, his main research interests include: Network security analysis, side-channel attack analysis.

RC4[23][24] Trivium[25][26] and so on.

Camellia is a 128-bit block cipher jointly developed by NTT and Mitsubishi Electric Corporation in 2000[28]. It is chosen as a recommended algorithm by the NESSIE (New European Schemes for Signatures, Integrity and Encryption) project in 2003[29] and is certified as the IETF (Internet Engineering Task Force) standard cipher for XML security URIs, SSL/TLS cipher suites and IPsec in 2005[30][31][32]. In March 2009, Camellia is integrated into the OPENSSL-1.0.0-beta1[32], which is the most widespread cryptographic library of the world. The first and only fault attack on Camellia as we have known is proposed by ZHOU Yongbin et. al. [16], they inject single byte fault to the 18th ,17th ,16th, 15th round of Camellia encryption left registers, finally, under ideal conditions, Camellia-128 key can be recovered with 64 faulty ciphertexts, and Camellia-192/256 key can be retrieved with 96 faulty ciphertexts. Their attack assume that only full 8 bytes of one Camellia round left registers have been injected fault twice, an equivalent subkey can be retrieved, thus at least 16 ciphertexts is needed to recover one round equivalent subkey. Recovering Camellia-128 needs at least 4 round equivalent subkeys, and Camellia-192/256 needs at least 6 round equivalent subkeys, so at least 64 and 96 ciphertexts are needed for recovering Camellia-128 and Camellia-192/256 key respectively. In practical, more than 16 ciphertexts are needed to recover one round equivalent subkey, so in all, the attacker needs far more than 64 and 96 ciphertexts to recover Camellia-128 and Camellia-192/256 key.

1.2 Our Contributions

The contributions of our work can be summarized as follows:

(1) Propose the basic differential fault attack model and present two DFA models for Feistel and SPN structure block ciphers.

The root of the DFA on block cipher with *S*-box lies in *S*-box itself, as long as the input and out differential value is known or analyzed by the adversary, it's quite easy for the adversary to recover the S-box input value, and combing certain analysis methods, the secret key can be recovered. We summarize the basic DFA model into the problem of computing the input and output differential value of the S-box lookups, and according to whether the input differential is known or unknown, present two DFA models for Feistel and SPN structures for block ciphers with S-box respectively. The proposed DFA model can be adapted into most of the block ciphers with S-box, such as AES, ARIA, CLEFIA , SMS4 etc.

(2) Present several improved differential fault attacks on Camellia encryption and extend these attacks to DFA on Camellia key schedule case.

Unlike attacks in [16] by injecting one byte fault into the r^{th} Camellia round left register to recover 1 byte of the r^{th} round equivalent key, we inject single byte fault into the r-1th Camellia round left register or the r-1th round key. Thanks to the Camellia permutation operation feature, one byte fault before the r-1th round S function can propagate 5 or 6 faulty bytes of the r^{th} round. We find out that, after solving 8 equations, it's possible to recover these 5 or 6 faulty input differential values of the r^{th} round, so 5 or 6 of the r^{th} round equivalent key bytes can be obtained by one time. Extensive experimentations have been performed on a PC and on average 4 faulty ciphertexts are enough to recover one round equivalent key, 16 faulty ciphertexts are enough to recover Camellia-128 key. Due to the non-regularity feature of the FL and FL^{-1} layer, it's quite complicated to recover the 13th round equivalent key by injecting fault into the 12th round before the last FL and FL^{-1} layer, so after injecting one byte fault to the 17th,16th,15th,14th,13th round, about 20 faults are enough to recover last 5 round equivalent key and limited candidates for 4 bytes of the 13th round equivalent key, in order to recover the full 13th round equivalent key, at most 12 extra times single byte fault are need to injected into the 13th round, the total fault number for recovering Camellia-192/256 key is about 32. If attacking the Camellia without FL and FL^{-1} layer, our attack needs about 24 faulty ciphertexts to recover full Camellia-192/256 key. It's clear to see that our attack needs far less faulty ciphertexts than [16], one byte fault before the r-1th round permutation is usually related with 5 or 6 r^{th} round equivalent key by DFA, and about random 4 faulty bytes can affect each of the 8 r^{th} round equivalent key bytes twice. This is much more effective than [16], which need to inject each of the 8 bytes before the r^{th} permutation for strictly twice. Table 1 demonstrates the improvement of our attacks over previous works.

Table 1. Improvement of our attacks over previous Camellia DFA work

Attack	Camellia	Fault Type	Fault Location	FL/FL^{-1}	Fault No
[16]	Camellia-128	18th,17th,16th,15th encryption round	$L_{17}, L_{16}, L_{15}, L_{14}$	x /√	64
[16]	Camellia-192/256	18th,17th,16th,15th,14th,13th encryption round	$L_{17}, L_{16}, L_{15}, L_{14}, L_{13}, L_{12}$	x /√	96
Section 4.3	Camellia-128	17th,16th,15th,14th encryption round	$L_{16}, L_{15}, L_{14}, L_{13}$	x /√	16
Section 4.4	Camellia-192/256	17th,16th,15th ,14th,13th,12th encryption round	L_{16}, L_{15}, L_{14} , L_{13}, L_{12}, L_{11}	x	24
Section 4.4	Camellia-192/256	17th,16th,15th,14th,13th encryption round	$L_{16}, L_{15}, L_{14}, L_{13}, L_{12}$	√	32
Section 4.5	Camellia-128	Camellia key schedule	$k_{17}, k_{16}, k_{15}, k_{14}$	x /√	16
Section 4.5	Camellia-192/256	Camellia key schedule	$k_{17}, k_{16}, k_{15}, k_{14}, k_{13}, k_{12}$	x	24

(3) Analyze the contradictions between traditional cryptography and implementation attacks, discuss the state of the art and possible future directions of the differential fault attack on Block ciphers with S-boxes.

Firstly, we take Cache based attack (CBA), Differential side channel attack (DSCA), Differential fault attack (DFA) etc attacks as an example to demonstrate the contradictions between traditional cryptography design and implementation attacks, and point out how to solve this problem is still unknown and confused every cryptographist. Secondly, we analyze the state of the art of DFA and point out some possible future directions of DFA on block ciphers with S-box.

1.3 Organization

This paper is organized as follows: Section 2 briefly describes preliminaries on Fault attack and Camellia. Section 3 presents the basic DFA model and how it can be used into SPN and Feistel block ciphers. Based on the Feistel DFA model, Section 4 presents several improved attacks on Camellia encryption, and then extends these attacks into DFA on Camellia key schedule case. Section 5 discusses on the contradictions between traditional cipher design and implementation attacks, makes some analysis on the state of the art and future directions of the DFA on Block ciphers with S-boxes, Section 6 is the conclusion.

2 Preliminaries

2.1 Fault Analysis

Fault Analysis is an active attack against the implementation of security modules. In the context of cryptanalysis, fault analysis aims to disturb the computation of a cryptographic algorithm in such a way that an erroneous result is obtained. By applying mathematical cryptanalysis these erroneous results can be used to extract cryptographic key material.

Generally speaking, fault attack is composed with two steps: fault injection and fault exploitation.

1 Fault injection: It mainly focuses on how to inject faults into the cryptographic devices during the encryption or decryption process on a proper time and location, its implementation and effects strongly depend on the fault injection equipment. Table 2 demonstrates the fault types according to different classifications. How to induce the specific types of faults is, however, not covered in this paper. The fault used in this paper adopts byte oriented fault model by computer simulations.

Table 2. Fault types

Fault classification	Fault Type

Injection methods	Glitch effects(Variations in supply voltage, external clock);
	Temperature effects(Variations in temperature);
	Light effects(Photoelectric effect, White light, Laser);
	Magnetic and Ray Radiation effects(Cosmic, α-, β-, X-rays)

duration	transient faults, permanent faults and destructive faults
control on the fault location	No control; Loose control" (a selected variable can be targeted); Complete control" (selected bits can be targeted).
control on the timing	No control; Loose control (a fault is induced in a block of a few operations); Precise control (the exact time can be met).

the number of bits affected	Single faulty bit;
	Few faulty bits (e.g., a byte);
	Random number of faulty bits (bounded by the length of the affected variable)

2 Fault exploitation: It mainly focuses on how to make full use of the generated faults and use certain analysis methods to retrieve partial or full secret key, it depends on both the injected fault type and cipher algorithms. In most cases, fault exploitation usually combines traditionally cryptanalysis methods such as differential and collision methods. Classical fault exploitation methods include: differential fault analysis (DFA), floating fault analysis (FFA), collision fault analysis (CFA), and ineffective fault analysis (IFA). In this paper, we mainly focus on DFA methods.

2.2 Camellia

A full description of the Camellia cipher is provided in[27][28], but below is a brief description of the cipher’s properties that are utilized in this study.

Encryption Procedure:

Camellia is an iterated cipher. Camellia takes a 128-bit plaintext P as input, and has a total of N rounds, where N is 18 for Camellia-128, and 24 for Camellia-192/256. Camellia-128(192/256) requires 22(29) rounds of data processing composed of three main parts: an 18(24)-round Feistel structure, two (three) FL function and FL^{-1} function rounds are inserted every 6 rounds, and two input/output whitenings. In the first and last round, the 128-bit data block is XORed with 128-bit round keys. Before the data block is fed to the Feistel network, it is separated into two 64-bit data blocks. The left half goes into the F function together with the 64-bit round key and the output of the F function is XORed with the right half block. At the end of each round, the right and left half block will be exchanged. In the F function, the input 64-bit data is first XORed with the 64-bit round key and then grouped into eight 8-bit data blocks. All of them are separately input to 8 S-boxes.

Let r be the number of the rounds of Camellia. Let L_{r-1} and R_{r-1} be the left and the right halves of r^{th} round inputs, and k_r be r^{th} round subkey. The Feistel structure for Camellia encryption can be written as follows:

$$\begin{cases} L_r = R_r \oplus F(L_{r-1}, k_r) \\ R_r = L_{r-1} \end{cases} \tag{1}$$

The round function F of Camellia is defined as

$$\begin{aligned} &F : F_2^{64} \times F_2^{64} \to F_2^{64} \\ &(X_{(64)}, k_{(64)}) \to Y_{(64)} = P(S(X_{(64)} \oplus k_{(64)})) \end{aligned} \tag{2}$$

The S function comprises 8 S-boxes, and 4 different types S-boxes s_0, s_1, s_2, s_3 are used(where s_1, s_2, s_3 are variations of s_0). The P function {0,1}64→{0,1}64 maps input(z_0, z_1,…, z_7) to output (z_0', z_1',…, z_7').

$$\begin{aligned} &z'_0 = z_0 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_7, z'_4 = z_0 \oplus z_1 \oplus z_5 \oplus z_6 \oplus z_7 \\ &z'_1 = z_0 \oplus z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7, z'_5 = z_1 \oplus z_2 \oplus z_4 \oplus z_6 \oplus z_7 \\ &z'_2 = z_0 \oplus z_1 \oplus z_2 \oplus z_4 \oplus z_5 \oplus z_7, z'_6 = z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_7 \\ &z'_3 = z_1 \oplus z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6, z'_7 = z_0 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 \end{aligned} \tag{3}$$

In Camellia, the FL and FL^{-1} functions inserted every 6 rounds are used to provide non-regularity between the rounds so that the security of the cipher is increased and these two functions are similarly constructed by logical operations including AND, OR, XOR, and rotations.

Key Schedule:

Fig.1 shows the key schedule of Camellia. Two 128-bit variables K_L and K_R are defined as follows. For 128-bit keys, the 128-bit key K is used as K_L and K_R is 0. For 192-bit keys, the left 128-bit of the key K is used as K_L, and concatenation of the right 64-bit of K and the complement of the right 64-bit of K is used as K_R. For 256-bit keys, the left 128-bit of the key K is used as K_L and the right 128-bit of K is used as K_R. Two 128-bit variables K_A and K_B are generated from K_L and K_R. Note that K_B is used only if the length of the secret key is 192 or 256 bits. The 64-bit constants $\sum_i$ (i = 1, 2, …, 6) are used as "keys" in the Feistel network. The 64-bit sub-keys k_{wt}, k_u, and k_{lv} are generated from K_L, K_R, K_A, and K_B. The sub-keys are generated by rotating K_L, K_R, K_A, and K_B for a specific number of bits and taking the left or right-half of them.

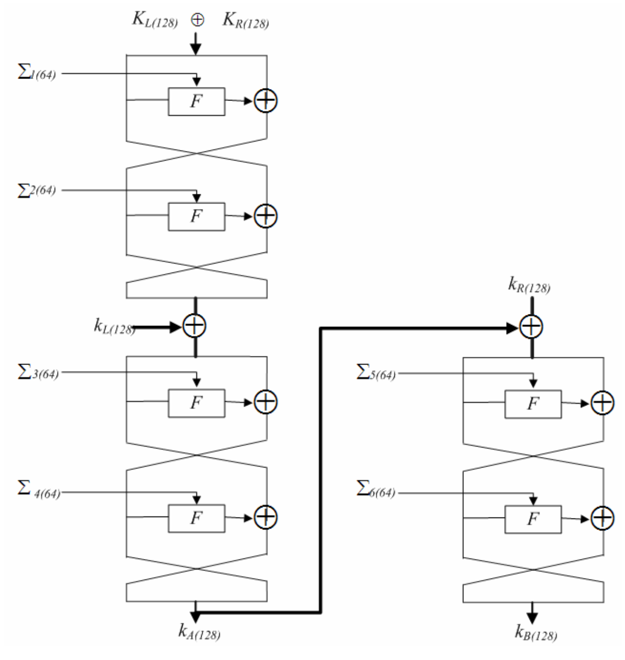


Fig.1 key schedule of Camellia

3 DFA Attack Model

3.1 Basic DFA Model

A lot of work have done on differential fault attacks, but seldom of them have explained the deeply roots of differential fault attack on block cipher with S-box. In this section, we try to analyze differential fault attack by a more straight and simple way.

Usually, the block cipher is composed with S function and P function. In differential fault attacks, the adversary usually injects a single byte fault before the final S function, one state byte a is changed into a*, $\Delta a = a \oplus a$*, the differential value Δa can be either known or unknown, also, the adversary can get the output differential value Δc. So, it always has the following formula:

$$S[a] \oplus S[a \oplus \Delta a] = \Delta c \tag{4}$$

Usually, the output of the S function $S[]$ usually has an extra output whitenings by Xored the last round key K_l to generate the ciphertexts C. As C is known, once a or $S[a]$ is known, K_l can be easily recovered.

According to whether Δa is known or unknown, there are two subcases: Δa is known or unknown, and based on the solutions to these two cases, two DFA models for Feistel and SPN block ciphers can be generated.

3.2 Feistel Structure Block Cipher DFA Model

If Δa is known, this case is usually related with Feistel structure block cipher. Fig.2 is the Feistel structure block cipher fault attack model, if we inject one byte fault Δa into $L_{r\text{-}1}$, due to the feature of Feistel structure, the differential value of $L_{r\text{-}1}$ is not changed, so after analyzing the differential value of C_L, we can get Δa, after analyzing the differential value of C_R, we can get the output differential value Δc.

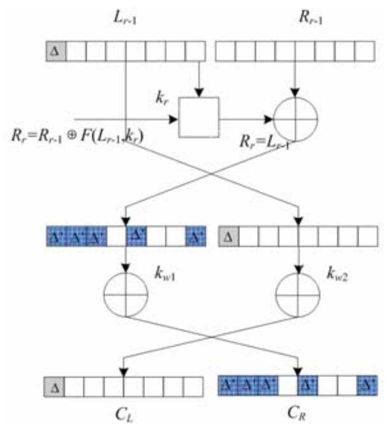
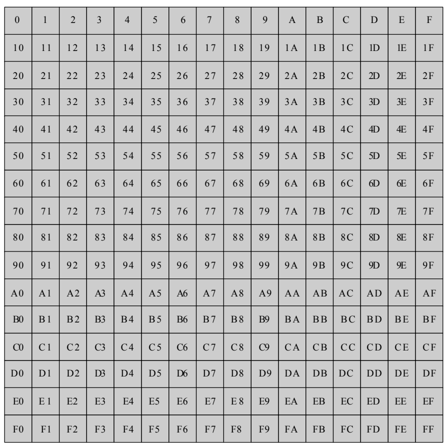


Fig.2 Feistel structure block cipher fault model

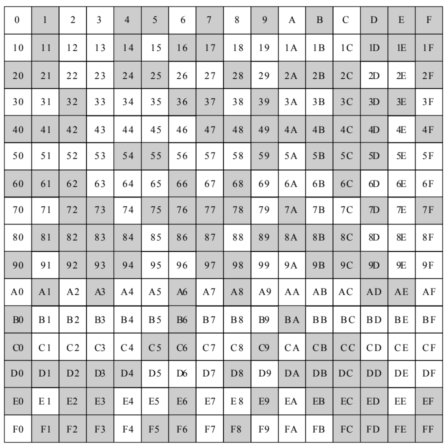
The only unknown variable is a, if we input every possible value of a into formula (4), we can get the possible candidates of a, usually, these candidates number is limited. Fig.3 is the S-box and differential S-box (Δ=1) of Camellia sorted descending, the gray square denotes candidates of S-box, and the white square denotes the impossible candidates of S-box. It's clear to see that the Camellia S-box S has covered with every distinct candidate value from 0x00 to 0xff (total number is 256), every candidate is used only once. However, when it comes to the differential S-box S'(S'[i]=S[i] ⊕ S[i ⊕ Δ], Δ=0x01), the new S-box S' is not perfect as S, S' hasn't covered with every distinct candidate value from 0x00 to 0xff(total number is 127), usually every possible candidate of S' is used twice or more. As to Camellia differential S-box, every possible candidate of S' is used about twice, so after we input every possible value of a into formula (4), usually, we can get 2 candidates of a, which means that we can also get 2 candidates for Camellia equivalent key.

S-box distributions



0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

differential S-box distributions(△=1)



0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

Fig.3 the S box and differential S box (Δ=1) of Camellia sorted descending

3.3 SPN Block Cipher DFA Model

If Δa is unknown, this case is usually related with SPN structure block cipher. Fig.4 is the SPN structure block cipher fault model. Unlike Feistel block cipher, when adversary injects one byte fault before the last permutation layer in the r-1th round, the output differential value is known but the input differential value is unknown. In order to recover a, the input differential value has to be guessed. Suppose Δa is a 8-bit non-zero value, which has 255 candidates. From Fig.3 we can known that the 7 output differential value should relate with the same differential S-box, if these 7 output values are not in the differential S-box, we can eliminate Δ=1, using this technique, we can get limited candidates of Δa (usually 2 for 7 output differential value), then case 2 can even be transferred into Case 1(Δa is known), finally, a can be recovered very efficiently.

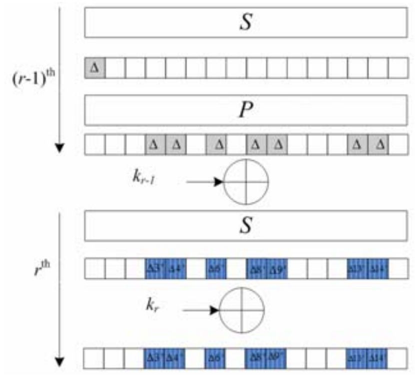


Fig.4 SPN structure block cipher fault model

4 Improved DFA on Camellia

4.1 Basic assumptions and Notations

Assumptions:

(1) Only one byte random fault is induced into the memory registers storing the intermediate results in one fault induction. Considering that the round function of Camellia takes only the left-half of the intermediate values as input, only the random single-byte fault occurred in the memory registers of the left-half input needs to be taken into account. Notice that the attacker knows neither the location nor the concrete value of the fault.

(2) For any one plaintext adaptively selected, two different ciphertexts under the control of the same secret key are available, the right ciphertext and the faulty one. This assumption is realistic in practice, for considering the scenarios under which an adaptive chosen message attack is mounted.

(3) The faulty ciphertexts of the required type are presumably available. How to induce the specific fault is not covered in this paper, since this is not the main concern of our paper. Nevertheless, the attacker should be able to identify the required faulty ciphertexts from a mass of faulty ciphertexts and discard those with faults occurring at a wrong timing.

(4) Only one user key is used during one successfully attack.

Notations:

(1) K_r: The equivalent subkey for rth round, is the exclusive OR of the half of the post-whitening subkey and rth subkey specified by Camellia algorithm specification, on the path from the corresponding half of the ciphertext to the input of the S-box of rth round, along the encryption flow in reverse order. In case of Camellia-128, for example, the equivalent subkey for 18th round is $K_{18}=k_{18}\oplus k_{w3}$, $K_{17}=k_{17}\oplus k_{w4}$ for 17th round, $K_{16}=k_{16}\oplus k_{w3}$ for 16th round, $K_{15}=k_{15}\oplus k_{w4}$ for 15th round, $K_{14}=k_{14}\oplus k_{w3}$ for 14th round, and $K_{13}=k_{13}\oplus k_{w4}$ for 13th round..

(2) $L_{r\text{-}1}$, $\mathrm{R}_{r\text{-}1}$: the 64-bit left and right halves of the rth round inputs.

(3) k_r, K_r: the 64-bit rth round subkey and equivalent subkey.

(4) $\Delta IL_r^{\,i}$, $\Delta IR_r^{\,i}$: the ith byte of the rth round left and right half input differential value.($i\in[0,7]$)

(5) ΔOLr^i, ΔORr^i: the ith byte of the rth round output differential value.($i\in[0,7]$)

(6) $\Delta S_r^{\,i}$, $\Delta P_r^{\,i}$: the ith byte of the rth round S function and P function output differential value.($i\in[0,7]$)

(7) ΔCL^i, ΔCR^i: the ith byte of the left and right half ciphertext differential value.($i\in[0,7]$)

(8) fault: If not specially stating, fault denotes the non-zero differential value besides the faulty ciphertext.

4.2 Former DFA on Camellia Encryption

ZHOU's attack[16] is a generic attack based on model in Section 3.2. It's main idea is to inject a single byte fault on $L_{r\text{-}1}$ and use DFA model of Section 3.2 to recover K_r. Specifically speaking, just take recovering K_{18} as an example, the adversary induces one byte fault ΔIL_{18} to L_{17}, then after the S function, the input differential fault ΔIL_{18} is transferred into single byte fault ΔS_{18}, then after the P function, 5 or 6 bytes of the output ΔP_{18} have the same fault as one byte of ΔS_{18}, after the final swap and exclusive OR of k_{w3} and k_{w4}, the ΔCL equals ΔIL_{18} and also is the input S function differential value, the ΔCR equals ΔP_{18} and also is the output S function differential value, by applying DFA methods in Section 3.2, the adversary can recover $L_{17}\oplus k_{18}$, as $L_{17}\oplus k_{w3}=C_L$, finally, the 18th equivalent subkey K_{18} can be recovered. Note that one single byte fault can only recover one K_{18} byte from 256 to 2-4 candidates, and two times of the same single byte fault can recover one correct K_{18} byte, so at least 16 faults are needed to recover K_{18}. By applying this method to 17th, 16th, 15th, the adversary can recover other three equivalent subkeys, and combing the key reversion techniques, the initial key can be recovered and verified.

4.3 Improved DFA on Camellia-128 Encryption

1 Main idea

The main idea of our DFA on Camellia is as follows:

(1) Induce one byte fault into the r-1th Camellia encryption round, and deduce equivalent subkey K_r.

(a) Choose any one plaintext P, and obtain the corresponding correct ciphertext C.

(b) Disturb the encryption of P until one byte random fault is successfully induced into only the input $L_{r\text{-}2}$ of the r-1th round, and obtain the corresponding faulty ciphertext C*.

(c) Deduce several bytes (5-6 bytes) of K_r using differential analysis technique.

(d) Repeat the above steps, until all bytes of K_r are recovered.

(2) Induce one byte fault into the r-2th Camellia encryption round, and deduce $K_{r\text{-}1}$.

(a) Choose any one plaintext P, and obtain the corresponding correct ciphertext C.

(b) Disturb the encryption of P until one byte random fault is successfully induced into only the input $L_{r\text{-}3}$ of the r-2th round, and obtain the corresponding faulty ciphertext C*.

(c) Deduce several bytes(5-6 bytes) of $K_{r\text{-}1}$ using differential analysis technique.

(d) Repeat the above steps, until all bytes of $K_{r\text{-}1}$ are recovered.

(3) Proceed in the same way and attack, in turn, the encryption of rounds r-3, r-4…, and deduce the equivalent subkeys $K_{r\text{-}2}$, $K_{r\text{-}3}$…., accordingly.

(4) Recover Camellia-128 key by analyze $K_{r\text{-}3}$, $K_{r\text{-}2}$, $K_{r\text{-}1}$,K_r by key reversion techniques, Camellia-192/256 key by analyze $K_{r\text{-}5}$, $K_{r\text{-}4}$,$K_{r\text{-}3}$,$K_{r\text{-}2}$,$K_{r\text{-}1}$, K_r.

(5) Verify the correctness of the recovered Camellia key.

2 Attack the 17th round to recover K_{18}.

Attacking the 17 round to recover K_{18} is depicted in Fig.5, specific attacking procedures are as follows:

(1) Choose randomly plaintext P and obtain the correct ciphertext C under the secret key K.

(2) Induce one random byte fault ΔIL_{17} into L_{16}, and obtain the faulty ciphertext C*.

The faulty propagate procedure is as follows:

(a) The 17th round: When one random single byte fault ΔIL_{17} is injected into L_{16}, suppose the faulty index is 0, so $\Delta IL_{17}{}^0 \neq 0$, $\Delta IL_{17}{}^i = 0$ ($i \in [1,7]$) then after the 17th round S function, the fault becomes $\Delta S_{17}{}^0$, after the 17th round P function, 5 bytes of fault were propagated ($\Delta P_{17}{}^0$, $\Delta P_{17}{}^1$, $\Delta P_{17}{}^2$, $\Delta P_{17}{}^4$, $\Delta P_{17}{}^7$ in equitation (5)), but all of the 5 faulty values are equal to $\Delta S_{17}{}^0$, after the exclusive OR of R_{16}, the left output differential equals ΔP_{17} (5 faulty bytes), and the right output differential equals ΔIL_{17} (1 faulty byte $\Delta IL_{17}{}^0$).

$$\begin{aligned}
&\Delta P_{17}{}^0 = \Delta S_{17}{}^0 \oplus \Delta S_{17}{}^2 \oplus \Delta S_{17}{}^3 \oplus \Delta S_{17}{}^5 \oplus \Delta S_{17}{}^6 \oplus \Delta S_{17}{}^7, \Delta P_{17}{}^4 = \Delta S_{17}{}^0 \oplus \Delta S_{17}{}^1 \oplus \Delta S_{17}{}^5 \oplus \Delta S_{17}{}^6 \oplus \Delta S_{17}{}^7 \\
&\Delta P_{17}{}^1 = \Delta S_{17}{}^0 \oplus \Delta S_{17}{}^1 \oplus \Delta S_{17}{}^3 \oplus \Delta S_{17}{}^4 \oplus \Delta S_{17}{}^6 \oplus \Delta S_{17}{}^7, \Delta P_{17}{}^5 = \Delta S_{17}{}^1 \oplus \Delta S_{17}{}^2 \oplus \Delta S_{17}{}^4 \oplus \Delta S_{17}{}^6 \oplus \Delta S_{17}{}^7 \\
&\Delta P_{17}{}^2 = \Delta S_{17}{}^0 \oplus \Delta S_{17}{}^1 \oplus \Delta S_{17}{}^2 \oplus \Delta S_{17}{}^4 \oplus \Delta S_{17}{}^5 \oplus \Delta S_{17}{}^7, \Delta P_{17}{}^6 = \Delta S_{17}{}^2 \oplus \Delta S_{17}{}^3 \oplus \Delta S_{17}{}^4 \oplus \Delta S_{17}{}^5 \oplus \Delta S_{17}{}^7 \\
&\Delta P_{173} = \Delta S_{17}{}^1 \oplus \Delta S_{17}{}^2 \oplus \Delta S_{17}{}^3 \oplus \Delta S_{17}{}^4 \oplus \Delta S_{17}{}^5 \oplus \Delta S_{17}{}^6, \Delta P_{17}{}^7 = \Delta S_{17}{}^0 \oplus \Delta S_{17}{}^3 \oplus \Delta S_{17}{}^4 \oplus \Delta S_{17}{}^5 \oplus \Delta S_{17}{}^6
\end{aligned} \tag{5}$$

(b) The 18th round: ΔIL_{18} has the same 5 faulty bytes equal to $\Delta S_{17}{}^0$, ΔIR_{18} has only one faulty byte $\Delta IL_{17}{}^0$. After the 18th round S function, the faulty values become 5 differential value $\Delta S_{18}{}^0$, $\Delta S_{18}{}^1$, $\Delta S_{18}{}^2$, $\Delta S_{18}{}^4$, $\Delta S_{18}{}^7$, after the 18th round P function, the faulty differential values become 8 distinct differential value ΔP_{18}, and after the exclusive OR of R_{17}, the faulty values become 8 distinct differential value ΔOL_{18}.

(c) The encryption output: After the 18th round, the output differential of C_L is equals ΔIL_{18}, has 5 faulty value equal to $\Delta S_{17}{}^0$, and the output differential of C_R is 8 distinct differential byte ΔOL_{18}.

(3) Deduce the fault location.

Different fault locations injected into L_{16} can generate different indices sets of the fault $C_{L,}$ the relationship between faulty byte index i of L_{16} and faulty bytes indices of C_L is shown in Table 3. According to this feature, the attacker can identify the fault location injected into L_{16} and choose corresponding further analysis strategies below.

Table 3. Induced one byte fault into $L_{16}{}^i$

Fault byte index i of L_{16}	Faulty bytes indices of C_L
i=0	0,1,2,4,7
i=1	1,2,3,4,5
i=2	0,2,3,5,6
i=3	0,1,3,6,7

Fault byte index *i* of L_{16}	Faulty bytes indices of C_L
i=4	1,2,3,5,6,7
i=5	0,2,3,4,6,7
i=6	0,1,3,4,5,7
i=7	0,1,2,4,5,6

(4) Deduce ΔS_{18} and $\Delta IL_{17}{}^{0}$.

It's easy to discover that 8 distinct differential value ΔOL_{18} is known to the attacker by analyzing $C_R \oplus C_R$*, and 8 $\Delta OL_{18}{}^{i}$ value is generated by $\Delta S_{18}{}^{0}$, $\Delta S_{18}{}^{1}$, $\Delta S_{18}{}^{2}$, $\Delta S_{18}{}^{4}$, $\Delta S_{18}{}^{7}$ and $\Delta IL_{17}{}^{0}$. All of this can generate 8 equations, it's quite simple to recover these 6 unknown differential value ($\Delta S_{18}{}^{0}$, $\Delta S_{18}{}^{1}$, $\Delta S_{18}{}^{2}$, $\Delta S_{18}{}^{4}$, $\Delta S_{18}{}^{7}$ and $\Delta IL_{17}{}^{0}$) by equation (6).

$$\left.\begin{array}{l}
\Delta OL_{18}{}^{0} = \Delta IL_{17}{}^{0} \oplus \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{1} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{2} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{183} = \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \\
\Delta OL_{18}{}^{4} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{5} = \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{6} = \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{7} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{4}
\end{array}\right\} \Rightarrow
\left.\begin{array}{l}
\Delta S_{18}{}^{0} = \Delta OL_{18}{}^{2} \oplus \Delta OL_{18}{}^{5} \\
\Delta S_{18}{}^{1} = \Delta OL_{18}{}^{5} \oplus \Delta OL_{18}{}^{6} \\
\Delta S_{18}{}^{2} = \Delta OL_{18}{}^{1} \oplus \Delta OL_{18}{}^{2} \\
\Delta S_{18}{}^{4} = \Delta OL_{18}{}^{1} \oplus \Delta OL_{18}{}^{4} \\
\Delta S_{18}{}^{7} = \Delta OL_{183} \oplus \Delta OL_{18}{}^{5} \\
\Delta IL_{17}{}^{0} = \Delta OL_{18}{}^{0} \oplus \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{7}
\end{array}\right\} \qquad (6)$$

(5) Recover 5 or 6 bytes of K_{18}.

From step (4) we can recover 5 output differential faulty bytes of the 18th *S* function, as the input differential fault is 5 equal faulty byte value $\Delta S_{17}{}^{0}$. By applying the general DFA model of Section 3.2, it's quite easy to recover 5 different *S* function input value, which can be expressed as $L_{17}{}^{i} \oplus k_{18}{}^{i}$ (*i*=0,1,2,4,7), as $L_{17}{}^{i} \oplus k_{w3}{}^{i} = C_L{}^{i}$, C_L is known to the attacker, so $k_{18}{}^{i} \oplus k_{w3}{}^{i}$ (*i*=0,1,2,4,7) can be recovered.

(6) Recover full 8 bytes of K_{18}.

Repeat step (1)-(5) to recover full 8 bytes of K_{18}. Note that after K_{18} is recovered, as every input 17th round S-box single byte fault ΔIL_{17} is recovered, and the output 17th round S-box differential value can be find out though ΔCL, using basic DFA on Camellia, the adversary can even recover several bytes of K_{17}.

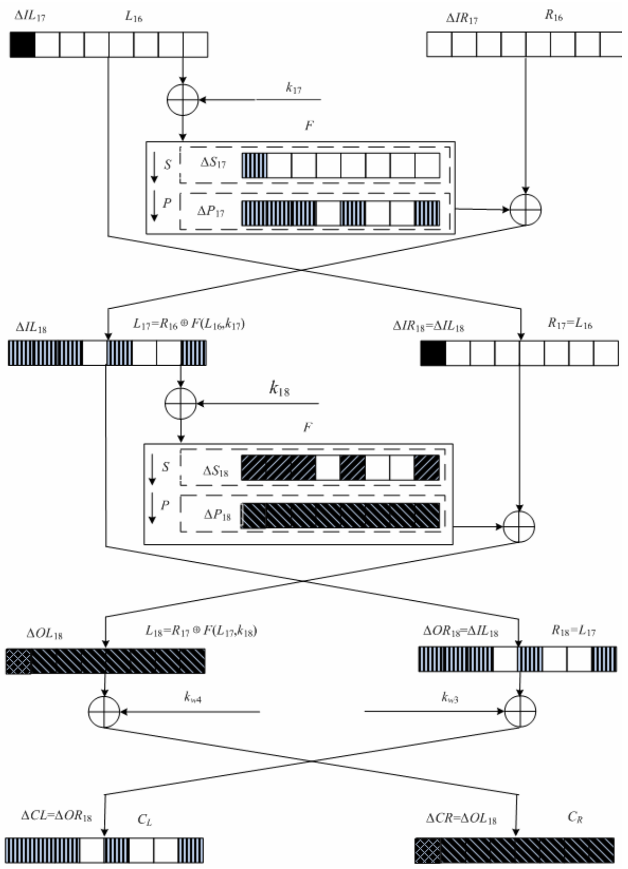


Fig.5 DFA model of attacking the Camellia 17th round to recover K_{18}

3 Attack the 16th round to recover K_{17}.

Attacking the 16 round to recover K_{17} is depicted in Fig.6. Specific attacking procedures are as follows:

(1) Choose randomly plaintext P and obtain the correct ciphertext C under the secret key K.

(2) Induce one random single byte fault ΔIL_{16} into L_{15}, and obtain the faulty ciphertext C*.

The faulty propagate procedure is as follows:

(a) The 16th round: When one random single byte fault $\Delta IL_{16}{}^{0}$ is injected into L_{15}, then after the 16th round S function, the fault becomes $\Delta S_{16}{}^{0}$, after the 16th round P function, 5 bytes of fault are propagated ($\Delta P_{16}{}^{0}$, $\Delta P_{16}{}^{1}$, $\Delta P_{16}{}^{2}$, $\Delta P_{16}{}^{4}$, $\Delta P_{16}{}^{7}$), but all of the 5 faulty values are equal to $\Delta S_{16}{}^{0}$, after the exclusive OR of R_{15}, the left output differential equals is ΔP_{16} (5 faulty bytes), and the right output differential is ΔIL_{16} (1 faulty byte $\Delta IL_{16}{}^{0}$).

(b) The 17th round: ΔIL_{17} has the same 5 faulty bytes equal to $\Delta S_{16}{}^{0}$, ΔIR_{17} has only one faulty byte $\Delta IL_{16}{}^{0}$. After the 17th round S function, the faulty values become 5 differential value $\Delta S_{17}{}^{0}$, $\Delta S_{17}{}^{1}$, $\Delta S_{17}{}^{2}$, $\Delta S_{17}{}^{4}$, $\Delta S_{17}{}^{7}$, after the 17th round P function, the faulty differential values become 8 distinct differential value ΔP_{17}, and after the exclusive OR of R_{16}, the faulty values become 8 distinct differential value ΔOL_{17}.

(b) The 18th round: ΔIL_{18} is equal to 8 distinct differential byte value ΔOL_{17} , ΔIR_{18} is equal to ΔIL_{17}(the same 5 faulty bytes equal to $\Delta S_{16}{}^{0}$). After the 18th round S function, the faulty values become 8 differential value ΔS_{18}, after the 18th round P function, the faulty differential values become 8 distinct differential value ΔP_{18}, and after the exclusive OR of R_{17}, the faulty values become 8 distinct differential value ΔOL_{18}.

(c) The encryption output: After the 18th round, the output differential of C_L is equals ΔIL_{18}, and the output differential of C_R is equal to ΔOL_{18}.

(3) Deduce ΔIL_{17}.

It's easy to see that ΔIL_{17} was equal to ΔP_{18} Xor-ed with ΔOL_{18}. ΔOL_{18} is equal to ΔCR and known to the attacker, all we need to do is to compute ΔP_{18}. In order to compute ΔP_{18}, we need to compute ΔS_{18}, as C_L, C_L* is known, K_{18} is recovered by step 2, ΔS_{18} can be obtained by equation (7), and then ΔP_{18} can be obtained by equation (8).

$$\Delta S_{18}{}^{i}=S[C_L{}^{i} \oplus K_{18}{}^{i}] \oplus S[C_L{}^{*i} \oplus K_{18}{}^{i}] \tag{7}$$

$$\begin{aligned}
\Delta P_{18}{}^{0} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7}, \Delta P_{184} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7} \\
\Delta P_{18}{}^{1} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7}, \Delta P_{185} = \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7} \\
\Delta P_{18}{}^{2} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{7}, \Delta P_{186} = \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{7} \\
\Delta P_{18}{}^{3} &= \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6}, \Delta P_{187} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6}
\end{aligned} \tag{8}$$

(4) Deduce the fault location.

Different fault locations injected into L_{15} can generate different indices sets of the fault $\Delta IL_{17,}$ the relationship between fault byte index i of L_{15} and Faulty byte indices of ΔIL_{17} is identical with Table 3. According to this feature, the attacker can identify the fault location injected into L_{15} and choose corresponding further analysis strategies below.

(5) Deduce ΔS_{17} and $\Delta IL_{16}{}^{0}$.

From (3) we can recover 5 input differential faulty bytes of the 17th S function ΔIL_{17}, in order to recover the 5 differential S function input value, we also need to recover ΔS_{17}. In order to recover ΔS_{17} and $\Delta IL_{16}{}^{0}$, we need to know about ΔIL_{18}, it's easy to discover that ΔIL_{18} is also equal to the output differential of C_L. After solving the equation (9), ΔS_{17} and $\Delta IL_{16}{}^{0}$ can be recovered.

$$\left.\begin{aligned}
\Delta IL_{18}{}^{0} &= \Delta IL_{16}{}^{0} \oplus \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{7} \\
\Delta IL_{18}{}^{1} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{7} \\
\Delta IL_{18}{}^{2} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{7} \\
\Delta IL_{18}{}^{3} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \\
\Delta IL_{18}{}^{4} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{7} \\
\Delta IL_{18}{}^{5} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{7} \\
\Delta IL_{18}{}^{6} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{7} \\
\Delta IL_{18}{}^{7} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{4}
\end{aligned}\right\} \Rightarrow \left.\begin{aligned}
\Delta S_{17}{}^{0} &= \Delta IL_{18}{}^{2} \oplus \Delta IL_{18}{}^{5} \\
\Delta S_{17}{}^{1} &= \Delta IL_{18}{}^{5} \oplus \Delta IL_{18}{}^{6} \\
\Delta S_{17}{}^{2} &= \Delta IL_{18}{}^{1} \oplus \Delta IL_{18}{}^{2} \\
\Delta S_{17}{}^{4} &= \Delta IL_{18}{}^{1} \oplus \Delta IL_{18}{}^{4} \\
\Delta S_{17}{}^{7} &= \Delta IL_{18}{}^{3} \oplus \Delta IL_{18}{}^{5} \\
\Delta IL_{16}{}^{0} &= \Delta IL_{18}{}^{0} \oplus \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{7}
\end{aligned}\right\} \tag{9}$$

(6) Recover 5 or 6 bytes of K_{17}.

As the input and output differential of the 17th S function is known, by applying the general DFA model of Section 3.2, it's easy to recover $L_{16}{}^{i} \oplus k_{17}{}^{i}$ (i=0,1,2,4,7), as $L_{16} \oplus P_{18} \oplus kw_4 = C_R$, $S_{18}=S[C_L \oplus K_{18}]$, P_{18} denotes the 18th round P function output, it can be computed by S_{18}, so $k_{17}{}^{i} \oplus kw_4{}^{i}$, which is also $K_{17}{}^{i}$(i=0,1,2,4,7) can be recovered.

(7) Recover full 8 bytes of K_{17}.

Repeat step (1)-(6) to recover full 8 bytes of K_{17}. Note that after K_{17} is recovered, as every input 16th round S-box single byte fault ΔIL_{16} is recovered, and the output 16th round S-box differential value ΔS_{16} (equals ΔIL_{17}) can be recovered, using basic DFA on Camellia, the adversary can even recover several bytes of K_{16}.

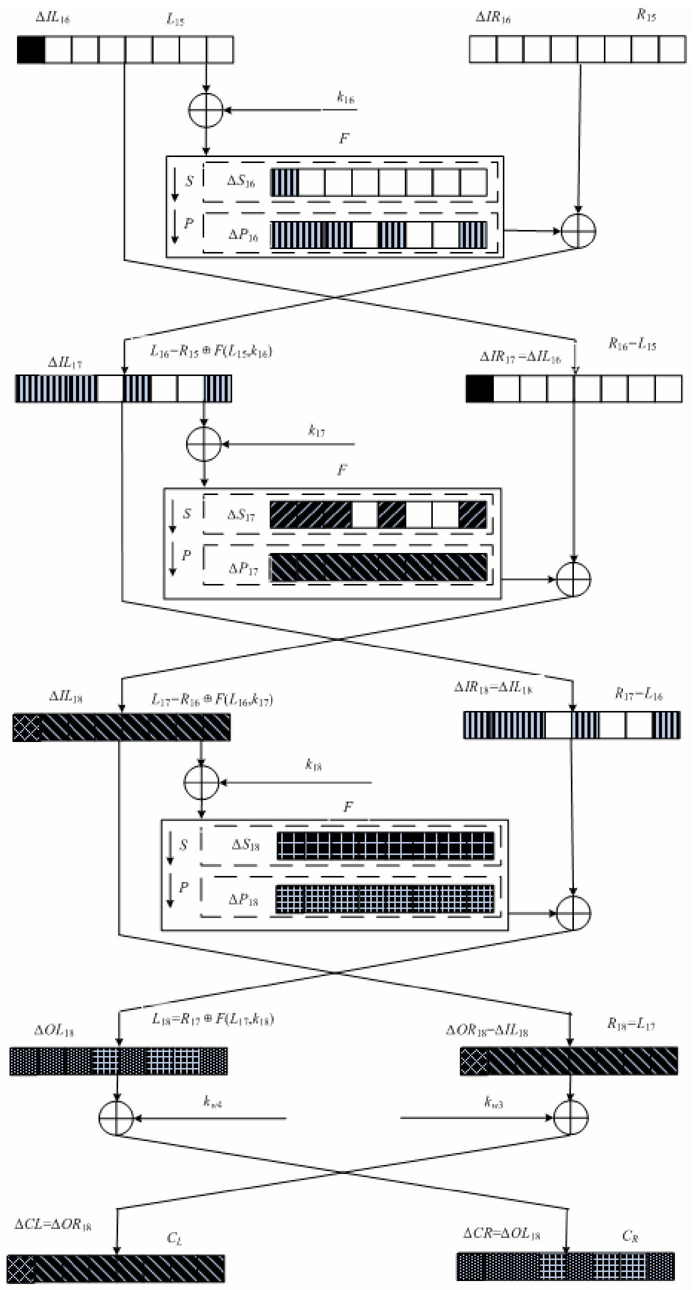


Fig.6 DFA model of attacking the Camellia 16th round to recover K_{17}

4 Attack the 15th round to recover K_{16}, Attack the 14th round to recover K_{15}.

Proceed in the same way as step 3 and step 4 above to recover K_{15} and K_{16}. Note that the key point is to find out input and output differential of the rth S-box, apply the DFA model of Section 3.2 to recover K_r.

5 Retrieve the initial Camellia-128 key.

From step 1 to 4, we can recover K_{15}, K_{16}, K_{17}, K_{18}. Learned from Camellia Specification[27], K_{15}, K_{16}, K_{17}, K_{18} can be expressed as:

$$\left.\begin{aligned} K_{18} &= (K_A <<< 111)_L \oplus (K_L <<< 111)_R \\ K_{17} &= (K_A <<< 111)_R \oplus (K_L <<< 111)_L \\ K_{16} &= (K_A <<< 111)_L \oplus (K_A <<< 94)_R \\ K_{15} &= (K_A <<< 111)_R \oplus (K_A <<< 94)_L \end{aligned}\right\} \Rightarrow \left.\begin{aligned} K_{18} \,\|\, K_{17} &= (K_A <<< 111) \oplus (K_L <<< 47) \\ K_{16} \,\|\, K_{15} &= (K_A <<< 111) \oplus (K_A <<< 30) \end{aligned}\right\}$$
$$\Rightarrow \left.\begin{aligned} (K_{16} \,\|\, K_{15}) <<< 17 &= K_A \oplus (K_A <<< 47) \\ K_L &= ((K_{18} \,\|\, K_{17}) >>> 47) \oplus (K_A <<< 64) \end{aligned}\right\} \tag{10}$$

From equation (10), we can get the value of $K_A \oplus (K_A <<<47)$, this is enough for us to recover K_A. Specific analysis method is as follows:

```
$K_A$ Searching Algorithm: Searching$K_A$($S_K$,$C$)
unsigned char $K_P$[128],cTemp
$S_K \leftarrow \emptyset$
Foreach $i$ from 0x00 to 0x01
{
  $K_P$[0] $\leftarrow i$
  Foreach $j$ from 0 to 127
  {
    cTemp $\leftarrow$ ($K_P$[(47*$j$)%128] ^ $C$[(47*$j$)%128])&0x01
    If($j$!=127)
      $K_P$[(47*($j$+1))%128] $\leftarrow$ cTemp;
    Else if($j$==127)
    {
      If(cTemp==$K_P$[0])
      Add $K_P$ to $S_K$
    }
  }
}
```

Using Algorithm above, we can get at most 2 candidates of K_A, sometimes even directly just get the unique value of K_A. After that, using equation (10) above, it's easy to recover K_L, K_L also has at most 2 candidates.

5 Verify the Camellia-128 key.

Using the plaintext and correct ciphertext, we can encrypt the plaintext with the prediction K_L and verify it's correctness by comparison of the ciphertext with the correct ciphertext. Finally, the correct initial Camellia-128 key can be verified.

4.4 Improved DFA on Camellia-192/256 Encryption

1 Recover K_{13}, K_{14}, K_{15}, K_{16}, K_{17}, K_{18}

The improved DFA on Camellia-192/256 method is much like Camellia-128. Note that if the attacked Camellia algorithm doesn't use *FL* and FL^{-1} layer, it's possible to recover K_{13}, K_{14}, K_{15}, K_{16}, K_{17}, K_{18} using the same methods in Section 4.3. But if the attacked Camellia algorithm has *FL* and FL^{-1} layer, using the same methods in Section 4.3, it's possible to recover K_{14}, K_{15}, K_{16}, K_{17}, K_{18}, K_{13} can be recovered by applying the DFA model of Section 3.2.

2 Retrieve the initial Camellia-192/256 key

$$\left.\begin{aligned} K_{18} &= (K_B <<< 111)_L \oplus (K_L <<< 111)_R \\ K_{17} &= (K_B <<< 111)_R \oplus (K_L <<< 111)_L \\ K_{16} &= (K_B <<< 111)_L \oplus (K_A <<< 94)_R \\ K_{15} &= (K_B <<< 111)_R \oplus (K_A <<< 94)_L \\ K_{14} &= (K_B <<< 111)_L \oplus (K_R <<< 94)_R \\ K_{13} &= (K_B <<< 111)_R \oplus (K_R <<< 94)_L \end{aligned}\right\} \Rightarrow \left.\begin{aligned} K_{18} \| K_{17} &= (K_B <<< 111) \oplus (K_L <<< 47) \\ K_{16} \| K_{15} &= (K_B <<< 111) \oplus (K_A <<< 30) \\ K_{14} \| K_{13} &= (K_B <<< 111) \oplus (K_R <<< 30) \end{aligned}\right\}$$
$$\left.\begin{aligned} & K_A \oplus K_R = (\ (K_{16} \| K_{15}) \oplus (K_{14} \| K_{13})\ ) >>> 30 \\ \Rightarrow\ & K_B = F(F((K_A \oplus K_R), \sum 5(64)), \sum 6(64)) \\ & K_L = (K_{18} \| K_{17}) >>> 47 \oplus (K_B <<< 64) \end{aligned}\right\} \tag{11}$$

From equation (11), it's clear to see that $K_A \oplus K_R$ can be recovered directly, as $K_A \oplus K_R$, $\sum_5$, $\sum_6$ is known, it's easy to compute K_B, finally, K_L can be retrieved, and the initial Camellia-192/256 key can be obtained.

4.5 DFA on Camellia Key Schedule

The DFA model in Section 4.3,4.4 can be easily adopted into attacking the Camellia key schedule procedure.

DFA on Camellia key schedule used in encryption is as follows:

(1) Induce one byte fault into k_{r-1} during the Camellia key schedule, after analyzing the faults propagated in the encryption procedure, deduce equivalent subkey K_r.

(a) Choose any one plaintext P, and obtain the corresponding correct ciphertext C.

(b) Induce one byte fault into $k_{r\text{-}1}$ during the Camellia key schedule, and obtain the corresponding faulty ciphertext C^* of encrypting P.

(c) Deduce several bytes (5-6 bytes) of K_r using differential analysis technique.

(d) Repeat the above steps, until all bytes of K_r are recovered.

(2) Induce one byte fault into $k_{r\text{-}2}$ during the Camellia key schedule, after analyzing the faults propagated in the encryption procedure, deduce $K_{r\text{-}1}$.

(a) Choose any one plaintext P, and obtain the corresponding correct ciphertext C.

(b) Induce one byte fault into $k_{r\text{-}2}$ during the Camellia key schedule, and obtain the corresponding faulty ciphertext C^* of encrypting P.

(c) Deduce several bytes(5-6 bytes) of $K_{r\text{-}1}$ using differential analysis technique.

(d) Repeat the above steps, until all bytes of $K_{r\text{-}1}$ are recovered.

(3) Proceed in the same way and attack, in turn, $k_{r\text{-}3}$, $k_{r\text{-}4}$…, after analyzing the faults propagated in the encryption procedure, deduce the equivalent subkeys $K_{r\text{-}2}$, $K_{r\text{-}3}$…., accordingly.

(4) Recover Camellia-128 key by analyze $K_{r\text{-}3}$, $K_{r\text{-}2}$, $K_{r\text{-}1}$,K_r by key reversion techniques, Camellia-192/256 key by analyze $K_{r\text{-}5}$, $K_{r\text{-}4}$,$K_{r\text{-}3}$,$K_{r\text{-}2}$,$K_{r\text{-}1}$, K_r.

(5) Verify the correctness of the recovered Camellia key.

This attack model of this Section is much alike Section 4.3,4.4, so we just take attacking the K_{18} as an example to explain it.

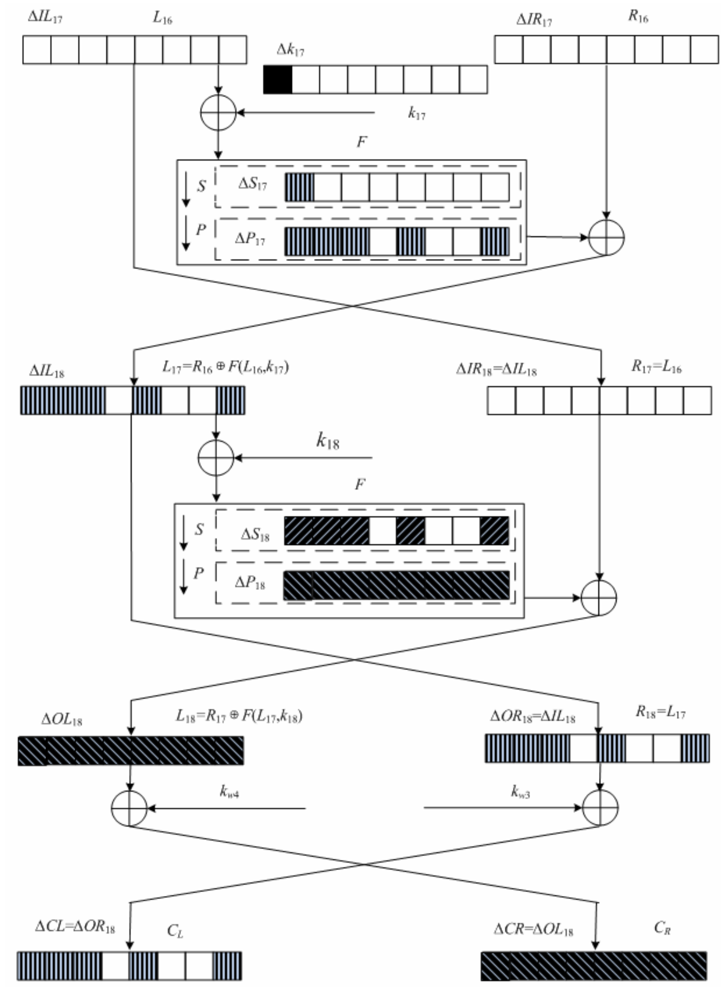


Fig.7 DFA model of attacking the Camellia key schedule to recover K_{18}

Suppose we can inject one byte random fault into the register storing k_{17}, then during the Camellia-128 encryption, the fault propagate procedure is almost the same as injecting one byte random fault into L_{16}, specific propagate procedure is shown in Fig.7. Note that the differences between Fig.7 and Fig.5 are as follows:

(1) The fault is injected into k_{17}, not L_{16}, but the fault propagated into ΔIL_{18} is the same as Fig.5, and no fault is in R_{17}.

(2) After the 18th round P function, the faulty differential values become 8 distinct differential value ΔP_{18}, and after the exclusive OR of R_{17}, the different faulty values are unchanged, which is different from Fig.5 (no fault is in R_{17}).

(3) The input and output differential of the 18th S function is the same as Fig.5, but we needn't to compute ΔIL_{17}.

$$\left.\begin{aligned}
\Delta OL_{18}{}^{0} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{1} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{2} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18^3} &= \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \\
\Delta OL_{18}{}^{4} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{5} &= \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{6} &= \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{7} \\
\Delta OL_{18}{}^{7} &= \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{4}
\end{aligned}\right\} \Rightarrow \left.\begin{aligned}
\Delta S_{18}{}^{0} &= \Delta OL_{18}{}^{2} \oplus \Delta OL_{18}{}^{5} \\
\Delta S_{18}{}^{1} &= \Delta OL_{18}{}^{5} \oplus \Delta OL_{18}{}^{6} \\
\Delta S_{18}{}^{2} &= \Delta OL_{18}{}^{1} \oplus \Delta OL_{18}{}^{2} \\
\Delta S_{18}{}^{4} &= \Delta OL_{18}{}^{1} \oplus \Delta OL_{18}{}^{4} \\
\Delta S_{18}{}^{7} &= \Delta OL_{18^3} \oplus \Delta OL_{18}{}^{5}
\end{aligned}\right\} \qquad (12)$$

(4) After attacking k_{17}, we can obtain recover K_{18}, but we can't recover any byte of K_{17} as in Section 4.3.

Proceed in the same way as step above to recover K_{17}, K_{16} and K_{15}, using key prediction and verification methods in Section 4.3, Camellia-128 key can be recovered very efficiently. If attacking Camellia-192/256 without FL/FL^{-1} layer, the adversary can proceed in the same way as step above to recover K_{18}, K_{17}, K_{15}, K_{16}, K_{15} and K_{14}. Using key prediction and verification methods in Section 4.4, Camellia-192/256 key can be recovered very efficiently.

4.6 Complexity Analysis

Due to the limit abilities of the attacker, it's very difficult to induce accurate and effective faults for DFA, so how many faulty ciphertexts are needed to crack the cipher is also very crucial. Next, we make a sketch of this complexity analysis.

$$M = \begin{vmatrix} 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1 \\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0 \\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1 \\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1 \\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1 \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1 \\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0 \end{vmatrix} \qquad (13)$$

According to the non-linear feature of the differential Camellia S-box distributions, it's not difficult to acquire that most of times, about two of the same one faulty S-box input byte can successfully recover one key byte. The relationship Matrix about fault byte index i of $L_{r\text{-}2}$ and recovered K_r byte indices of this paper is shown in equation (13), M_{ij}=1 denotes that it's a related key byte, else not, i denotes the fault byte index of $L_{r\text{-}2}$ and j denotes the index of K_r. It's clear to see that when $i \in [0,3]$, one fault in $L_{r\text{-}2}$ can relate with 5 bytes of K_r, and when $\mathrm{i} \in [4,7]$, one fault in $L_{r\text{-}2}$ can related with 6 bytes of K_r. At least 3 random faults can cover 8 bytes of K_r twice, so ideally speaking, at least 3 faulty pairs ciphertexts is enough to recovered K_r. The relation between fault number and success rate of recovering K_r is shown in Table 4. In practical, during the experiment, we find out that about 4 faults is enough to recover K_r.

Table 4. Fault Number and Success Rate of Recovering K_r

Fault Number	Success Count	Full Count	Success Rate
3	24	512	4.69%
4	1608	4096	39.26%
5	22880	32768	69.82%
6	226920	262144	86.56%
7	1977360	2097152	94.29%
8	16378376	16777216	97.62%

So, about 4 random faulty ciphertexts are enough to recover one round equivalent subkey, 16 ciphertext are enough to recover Camellia-128 key. Due to the non-regularity feature of the FL and FL^{-1} layer, it's quite complicated to recover K_{13} by injecting fault to L_{11}, so after injecting single byte fault to the 17th,16th,15th,14th,13th round, about 20 faults are enough to recover full bytes of K_{18}, K_{17}, K_{16}, K_{15}, K_{14}, K_{13}, and limited candidates for 4 bytes of K_{13}, in order to recover full K_{13} key bytes, we need to inject at most 12 extra times single byte fault to the 13th round, using the basic DFA Camellia methods in Section 4.2 to recover K_{13}. So the total fault count for recover Camellia-192/256 key is at most 32 faulty ciphertexts (in fact 24 is enough). If attacking the Camellia without FL and FL^{-1} layer, our attack needs about 24 faulty ciphertexts to recover full Camellia-192/256 key. Noting that it's impossible to recover K_r by injected fault into the k_r in the case of DFA on Camellia key schedule, it's impossible to recover K_{13} by injecting fault into k_{13}, so our DFA methods on Camellia key schedule can not recover K_{13} of Camellia with FL /FL^{-1} layer.

4.7 Experimental Results and Comparisons

We have implemented the experimental simulations of the attacks given in this paper. The simulations are written in Visual C++6.0 on Windows XP. Our simulations run on a personal computer (Athlon 64-bit 3000+ 1.81 GHz CPU and 1GB RAM) and successfully extract the full 128-bit Camellia key by using 16 faulty ciphertexts of Camellia with and without FL/FL^{-1} layer, the full 192/256-bit Camellia key by using 32 and 24 faulty ciphertexts of Camellia 192/256 with and without FL/FL^{-1} layer respectively. These experimental results strongly support the complexity analysis presented in Section 4.5. See the appendix B for data we used in one successful simulation DFA on Camellia-128 encryption procedure. It's clear to see that our DFA methods are much more effective than ZHOU's attack in [16], Our approach has the following properties:

(1) Firstly, the assumption of our proposed fault type is the same as [16], but the fault depth has been enhanced. That is, in the chosen plaintext attacks, some registers in one round can be induced one byte fault. To recover the r^{th} round equivalent subkey K_r, our proposed fault location is in L_{r-2}, while [16] shows that the first fault location is in L_{r-1}.

(2) Secondly, in the same assumptions as above, the faulty ciphertexts number of our proposed attacking methods is far less than [16](See Table 1). If [16] is firing two times to kill one bird, ours are firing two times to kill 5 or 6 birds.

(3) Thirdly, in the same assumptions above, the efficiency of our proposed attacking methods is higher than that in [16]. In order to recover Camellia-128 key, K_{18}, K_{17}, K_{16}, K_{15} are needed to be recovered one by one. Recovering K_{r-1} needs the precondition of successfully recover K_r. In [16], recover K_r needs to accurately inject into every index of L_{r-1} twice, so 16 faults needed is a rather ideal case. In the real condition, about 32 or even more random faults are needed in order to cover every index of L_{r-1} twice. The attack proposed in this paper needs far loose conditions, on average 4 random faults is enough to propagate the faults into every index of L_{r-1} twice in practical, thus are much more effective than [16].

(4) Lastly, our DFA methods on Camellia encryption procedure can be easily adapted into DFA on Camellia key schedule case, while [16] can not. It's impossible for [16] to inject fault into k_r to recover K_r(the adversary can not get the input differential value of the r^{th} S-box lookup), however, according to the analysis of the Section 4.5, it's quite easy to inject fault into k_{r-1} to recover K_r.

5 Discussions

5.1 Contradictions between traditional cryptography design and implementation attacks

In the cryptography design, cryptographists usually add more non-linear (S-box lookups) and complicated linear operations to prevent cipher algorithms from linear and differential attack and it indeed works well on traditional mathematical analysis. But when it comes to the cryptosystem implementation, it meets unprecedented challenges. It is just the non-linear part operations leaking more information on secret key. The S-box lookup, as one of the most important operations during the cipher algorithm design, and also is the most effective part to prevent traditional linear and differential attacks, usually, the input of the S-box is related with one plaintext\ciphertext byte, one initial key or subkey byte, and the elements of the S-box is open to the public. Next, we take the S-box lookup as an example and discuss the relationships between it and implementation analysis.

1 Cache based attack (CBA):

Cache hit and miss feature can affect the whole encryption time and the accessed Cache sets information of the cryptosystems, this can be utilized as timing driven and access driven Cache attacks. In timing driven attack, the adversary can use the whole encryption time to predict whether two times S-box lookup is Cache hit or miss, thus get the possible or impossible key byte candidates. In access driven attack, the adversary can use a spy process loading a L1 Cache size array to clear the Cache before the encryption, then trigger the cipher encrypt operations, after that, the spy process can gather the accessed Cache sets of the encryption process by measuring the time (Cache hit and Cache miss time are different) to reload each Cache block size array element. Combing the plaintext and ciphertext, the adversary can get the possible or impossible candidates for the encryption key and finally recover the whole encryption key. Note that it was just the frequent S-box lookup operations leading to Cache timing attacks.

2 Differential side channel attack (DSCA):

In differential side channel attack, the attacker gets several power consumption or electromagnetic emanation curves. As the S-box dictionary is known to the attacker, the adversary first divides the key search space to several bytes, then tries every possible value of each byte to predict one or bits of the S-box lookup results and get the possible hamming weight or distance, combing the real power or electromagnetic curves which is affected by the hamming weight or distance. Then, the correct key bytes can always has high coefficient than wrong key bytes, so the S-box lookups can also be used in DSCA to recover encryption key. In fact, beside S-box lookup operation, any operations in encryption with strong non-linear feature can be used in DSCA to recover encryption key.

3 Differential fault attack(DFA)

From Section 3 and the DFA on Camellia in Section 4, we can see that the root of the differential fault attack on block cipher with *S*-box lies in *S*-box itself. As long as the input differential and output differentia of S-box is known to the attacker, it's very easy to recover the input value of the *S*-box, which can be used for further analyzing and recovering of the encryption key. Current *S*-box is not perfect, the differential *S*-box doesn't cover all the candidates from 0x00 to 0xff, which leaks some information about the input differential once the adversary gets the output differential for SPN block ciphers, even if the differential S-box is prefect (cover all the candidates from 0x00 to 0xff), as the output differential and input differential value of the Feistel block ciphers is known to the adversary, it may become more easy to recover the S-box input value by just one fault, which make differential fault attack become more effective. Anyway, S-box makes differential fault attack on block cipher with S-box becomes possible.

From analysis above, we can safely come to the conclusion that there indeed exist great contradictions between traditional cipher design and implementation attacks, but how to solve this problem is still unknown and confused every cryptographist.

5.2 State of the art and the future directions of the DFA on Block ciphers with S-boxes

Analyzed from these published papers of DFA on Block ciphers with S-boxes (DES[6],3DES[8], AES[9][10][11][12][13][14][15], Camellia[16], ARIA [17], CLEFIA[18][19],SMS4[20][21][22]), it's easy to discover that all of them were based on DFA methods. In our opinion, all of the attacks above are based on the basic DFA model in Section 3.1, and according to the different structure of block ciphers and whether the input differential fault of the S-box lookup is known or unknown, all of the attacks above can be classified into Feistel structure DFA model in Section 3.2 and SPN structure model in Section 3.3 two kinds. The differences between these attacks depend highly on the specific attacked cipher algorithm design and the faults propagate features.

Based on the current development trend of DFA on block ciphers with S-box, and the analysis methods in other implementation attacks, we can analyze and predict the future possible directions of DFA:

(1) Broaden fault width

At the beginning of the DFA on DES, the fault width is 1 bit, this is because DES is a bit based block cipher. When DFA was first used on AES, the fault width is also 1 bit in [9], and after that, more and more DFA on AES and other block ciphers are based on one byte ([10][11][12][13][14][15][16][17][18][19][20][21][22]). We think that, on the precondition of identify faulty parts, broaden fault width to more bits even multi-bytes maybe one of the directions for DFA.

(2) Enhance fault depth

At the beginning, in DFA attacks on Feistel structure block cipher, the attacker usually attacks the r^{th} round (before the r^{th} round *P* function) to recover the r^{th} round key, such as [9][10][11][17] [20], and in DFA attacks on SPN structure block cipher, the attacker usually attacks the r-1^{th} round(before the r-1^{th} round *P* function) to recover the r^{th} round key such as [16][19]. With the development of DFA methods, the fault depths are enhanced, the attacker usually attacks the r-n^{th} (n>1) round (before the r-n^{th} round *P* function) to recover the r^{th} round key, such as [12][13][21][22].

(3) Combing fault analysis.

The first is combing other side channel attacks with fault attacks, such as power attacks, electromagnetic attacks with fault attacks, and the second is combing other traditional mathematical analysis methods with fault analysis, such as collision analysis, stochastic analysis methods.

6 Conclusion

In this paper we present the basic differential fault attack model and how it can be used into SPN and Feistel block ciphers, then take Camellia block cipher as an example, propose several improved differential fault attacks on Camellia encryption procedure and extend them into DFA on Camellia key schedule case. With the byte oriented fault model, only 16 ciphertexts are required to obtain the Camellia-128 key, 32, 24 ciphertexts are required to obtain both Camellia-192/256 with and without FL/FL^{-1} layer respectively. Our methods not only extend the fault location, but also improve the efficiency of fault injection and decrease the number of faulty ciphertexts. Besides, our attack model can be adapted into most of block cipher with S-boxes, such as AES, ARIA, CLEFIA , SMS4 etc. Further more, all the attacks described in this paper have been successfully put into experimental simulations on a personal computer and the experimental results effectively support the analysis and the arguments.

Acknowledgements

We are indebted to ZHENG Tian-ming, YANG Guang-kai for their interests and discussions about Camellia fault analysis methods, LUO Lan for her patientce and insightful advices on this paper, and the (anonymous) referees for their suggestions.

Appendix A Other 7 groups of Equations to recover Camellia Equivalent subkey

Table A-1 Induced a single byte fault into the $L_{r\text{-}2}{}^{i}$

Fault Byte Index	groups of Equations
i=1	$\left.\begin{aligned} \Delta IL_{18}{}^{0} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \\ \Delta IL_{18}{}^{1} &= \Delta IL_{16}{}^{1} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \\ \Delta IL_{18}{}^{2} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{5} \\ \Delta IL_{18}{}^{3} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{5} \\ \Delta IL_{18}{}^{4} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{5} \\ \Delta IL_{18}{}^{5} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \\ \Delta IL_{18}{}^{6} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{5} \\ \Delta IL_{18}{}^{7} &= \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{5} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} \Delta S_{17}{}^{1} &= \Delta IL_{18}{}^{3} \oplus \Delta IL_{18}{}^{6} \\ \Delta S_{17}{}^{2} &= \Delta IL_{18}{}^{6} \oplus \Delta IL_{18}{}^{7} \\ \Delta S_{17}{}^{3} &= \Delta IL_{18}{}^{2} \oplus \Delta IL_{18}{}^{3} \\ \Delta S_{17}{}^{4} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{6} \\ \Delta S_{17}{}^{5} &= \Delta IL_{18}{}^{2} \oplus \Delta IL_{18}{}^{5} \\ \Delta IL_{16}{}^{1} &= \Delta IL_{18}{}^{1} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \end{aligned}\right\}$
i=2	$\left.\begin{aligned} \Delta IL_{18}{}^{0} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{1} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{2} &= \Delta IL_{16}{}^{2} \oplus \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{5} \\ \Delta IL_{18}{}^{3} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{4} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{5} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{6} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \\ \Delta IL_{18}{}^{7} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} \Delta S_{17}{}^{0} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{3} \\ \Delta S_{17}{}^{2} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{7} \\ \Delta S_{17}{}^{3} &= \Delta IL_{18}{}^{4} \oplus \Delta IL_{18}{}^{7} \\ \Delta S_{17}{}^{5} &= \Delta IL_{18}{}^{1} \oplus \Delta IL_{18}{}^{7} \\ \Delta S_{17}{}^{6} &= \Delta IL_{18}{}^{1} \oplus \Delta IL_{18}{}^{7} \\ \Delta IL_{16}{}^{2} &= \Delta IL_{18}{}^{2} \oplus \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{5} \end{aligned}\right\}$
i=3	$\left.\begin{aligned} \Delta IL_{18}{}^{0} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{1} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{2} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{3} &= \Delta IL_{16}{}^{3} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{4} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{5} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{6} &= \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{7} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} \Delta S_{17}{}^{0} &= \Delta IL_{18}{}^{4} \oplus \Delta IL_{18}{}^{5} \\ \Delta S_{17}{}^{1} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{1} \\ \Delta S_{17}{}^{3} &= \Delta IL_{18}{}^{1} \oplus \Delta IL_{18}{}^{4} \\ \Delta S_{17}{}^{6} &= \Delta_{2}{}'''' \oplus \Delta IL_{18}{}^{4} \\ \Delta S_{17}{}^{7} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{7} \\ \Delta IL_{16}{}^{3} &= \Delta IL_{18}{}^{3} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \end{aligned}\right\}$
i=4	$\left.\begin{aligned} \Delta IL_{18}{}^{0} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{1} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{2} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{3} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{4} &= \Delta IL_{16}{}^{4} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{5} &= \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{6} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{7} &= \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} \Delta S_{17}{}^{6} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{6} \\ \Delta S_{17}{}^{5} &= \Delta IL_{18}{}^{2} \oplus \Delta IL_{18}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta S_{17}{}^{3} &= \Delta IL_{18}{}^{7} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta S_{17}{}^{2} &= \Delta IL_{18}{}^{1} \oplus \Delta\Delta IL_{18}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta S_{17}{}^{1} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \\ \Delta S_{17}{}^{7} &= \Delta IL_{18}{}^{7} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{16}{}^{4} &= \Delta IL_{18}{}^{4} \oplus \Delta S_{17}{}^{1} \oplus \Delta S_{17}{}^{5} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \end{aligned}\right\}$
i=5	$\left.\begin{aligned} \Delta IL_{18}{}^{0} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{1} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{2} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{3} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{6} \\ \Delta IL_{18}{}^{4} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{5} &= \Delta IL_{16}{}^{5} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{6} &= \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{18}{}^{7} &= \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{3} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{6} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} \Delta S_{17}{}^{7} &= \Delta IL_{18}{}^{1} \oplus \Delta IL_{18}{}^{7} \\ \Delta S_{17}{}^{6} &= \Delta IL_{18}{}^{3} \oplus \Delta IL_{18}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta S_{17}{}^{0} &= \Delta IL_{18}{}^{4} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta S_{17}{}^{4} &= \Delta IL_{18}{}^{0} \oplus \Delta IL_{18}{}^{3} \oplus \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{7} \\ \Delta S_{17}{}^{3} &= \Delta IL_{18}{}^{2} \oplus \Delta IL_{18}{}^{3} \oplus \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \\ \Delta S_{17}{}^{2} &= \Delta IL_{18}{}^{1} \oplus \Delta IL_{18}{}^{3} \oplus \Delta S_{17}{}^{0} \oplus \Delta S_{17}{}^{7} \\ \Delta IL_{16}{}^{5} &= \Delta IL_{18}{}^{5} \oplus \Delta S_{17}{}^{2} \oplus \Delta S_{17}{}^{4} \oplus \Delta S_{17}{}^{6} \oplus \Delta S_{17}{}^{7} \end{aligned}\right\}$

i=6	$\left.\begin{aligned} \Delta IL_{18}^{0} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{3} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{7} \\ \Delta IL_{18}^{1} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{3} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{7} \\ \Delta IL_{18}^{2} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{7} \\ \Delta IL_{18}^{3} &= \Delta S_{17}^{1} \oplus \Delta S_{17}^{3} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \\ \Delta IL_{18}^{4} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{7} \\ \Delta IL_{18}^{5} &= \Delta S_{17}^{1} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{7} \\ \Delta IL_{18}^{6} &= \Delta IL_{16}^{6} \oplus \Delta S_{17}^{3} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{7} \\ \Delta IL_{18}^{7} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{3} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} \Delta S_{17}^{4} &= \Delta IL_{18}^{2} \oplus IL_{18}^{4} \\ \Delta S_{17}^{7} &= \Delta IL_{18}^{0} \oplus \Delta IL_{18}^{7} \oplus \Delta S_{17}^{4} \\ \Delta S_{17}^{1} &= \Delta IL_{18}^{5} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{7} \\ \Delta S_{17}^{5} &= \Delta IL_{18}^{0} \oplus \Delta IL_{18}^{1} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{4} \\ \Delta S_{17}^{3} &= \Delta IL_{18}^{3} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \\ \Delta S_{17}^{0} &= \Delta IL_{18}^{4} \oplus \Delta IL_{18}^{1} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{7} \\ \Delta IL_{16}^{6} &= \Delta IL_{18}^{6} \oplus \Delta S_{17}^{3} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{7} \end{aligned}\right\}$
i=7	$\left.\begin{aligned} \Delta IL_{18}^{0} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{2} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{6} \\ \Delta IL_{18}^{1} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{6} \\ \Delta IL_{18}^{2} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{2} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \\ \Delta IL_{18}^{3} &= \Delta S_{17}^{1} \oplus \Delta S_{17}^{2} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{6} \\ \Delta IL_{18}^{4} &= \Delta S_{17}^{0} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{6} \\ \Delta IL_{18}^{5} &= \Delta S_{17}^{1} \oplus \Delta S_{17}^{2} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{6} \\ \Delta IL_{18}^{6} &= \Delta S_{17}^{2} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \\ \Delta IL_{18}^{7} &= \Delta IL_{16}^{7} \oplus \Delta S_{17}^{0} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{6} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} \Delta S_{17}^{5} &= \Delta IL_{18}^{3} \oplus IL_{18}^{5} \\ \Delta S_{17}^{4} &= \Delta IL_{18}^{1} \oplus \Delta IL_{18}^{4} \oplus \Delta S_{17}^{5} \\ \Delta S_{17}^{2} &= \Delta IL_{18}^{6} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \\ \Delta S_{17}^{1} &= \Delta IL_{18}^{0} \oplus \Delta IL_{18}^{1} \oplus \Delta S_{17}^{2} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \\ \Delta S_{17}^{6} &= \Delta IL_{18}^{1} \oplus \Delta IL_{18}^{2} \oplus \Delta S_{17}^{2} \oplus \Delta S_{17}^{5} \\ \Delta S_{17}^{0} &= \Delta IL_{18}^{4} \oplus \Delta S_{17}^{1} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{6} \\ \Delta IL_{16}^{7} &= \Delta IL_{18}^{7} \oplus \Delta S_{17}^{0} \oplus \Delta S_{17}^{4} \oplus \Delta S_{17}^{5} \oplus \Delta S_{17}^{6} \end{aligned}\right\}$

Appendix B Experimental data used in simulation attack

Due to the limit of space, only the experimental data used in simulation of attacking Camellia-128 encryption procedure are shown. One pair of the plaintext and the right ciphertext used to verify the equivalent subkey candidates is：

P = 2b 80 18 2f 4d c9 3d f1 44 91 30 7b 00 4c ec 96

C = ec e0 83 64 26 1e ed 0c 51 09 f9 3a 08 47 5d 45

Note that during the fault attack on the Camellia last 4 rounds, we adopt the same 4 plaintexts. The right pairs of ciphertexts used in the experimental simulation are given in Table B-1.

Table B-1 Pairs of ciphertexts used in our simulation DFA Experiment on Camellia-128

Camellia Round	Correct ciphertext	Faulty ciphertext
18th	ec e0 83 64 26 1e ed 0c 51 09 f9 3a 08 47 5d 45 e1 4b 34 60 2d 8d 93 ee fa 3f 3d 4a ed b4 52 62 4c 3f c0 66 bc 10 15 9f de eb f6 cd ca d4 d9 93 5d 1e 08 b7 a2 f7 29 8b 06 33 5a d4 20 6a 50 46	a8 a4 c7 64 62 5a a9 0c 6f 8e 0d 2f a7 b6 4e 8c 28 4b fd a9 e4 8d 5a 27 b0 59 fd e1 46 18 79 c3 3d 3f b1 17 cd 10 64 ee 46 a1 4e 72 a2 8a ea 57 5d 3b 2d 92 a2 d2 c ae c5 53 6e 79 38 28 60 cd
17th	ec e0 83 64 26 1e ed 0c 51 09 f9 3a 08 47 5d 45 e1 4b 34 60 2d 8d 93 ee fa 3f 3d 4a ed b4 52 62 4c 3f c0 66 bc 10 15 9f de eb f6 cd ca d4 d9 93 5d 1e 08 b7 a2 f7 29 8b 06 33 5a d4 20 6a 50 46	43 81 e9 49 50 60 a2 e7 d8 c6 9e 10 77 b6 e2 94 02 f3 9b 2c 22 62 30 19 2a 6b 9f ed 22 c2 95 d1 8b 55 d2 5a 6f 72 8a bf 29 2f cc 45 4c d8 ef 5f e6 8d 4c 67 3e 44 f1 e7 eb 44 ba 2f 74 64 4 d7
16th	ec e0 83 64 26 1e ed 0c 51 09 f9 3a 08 47 5d 45 e1 4b 34 60 2d 8d 93 ee fa 3f 3d 4a ed b4 52 62 4c 3f c0 66 bc 10 15 9f de eb f6 cd ca d4 d9 93 5d 1e 08 b7 a2 f7 29 8b 06 33 5a d4 20 6a 50 46	ba 9d 36 06 2f 24 d4 3b 3e 8b f5 65 5b 15 08 b7 38 00 2f cd bb d0 a4 41 6f 94 3d ee 92 ef 07 15 95 bd 71 6b 10 af 86 d4 ee dc 20 d2 cf c3 52 5e 3c 03 64 d6 24 a2 af a4 b0 f4 66 04 28 43 90 74
15th	ec e0 83 64 26 1e ed 0c 51 09 f9 3a 08 47 5d 45 e1 4b 34 60 2d 8d 93 ee fa 3f 3d 4a ed b4 52 62 4c 3f c0 66 bc 10 15 9f de eb f6 cd ca d4 d9 93 5d 1e 08 b7 a2 f7 29 8b 06 33 5a d4 20 6a 50 46	e5 2d 47 7e 96 85 e5 4a 8d d9 c4 7c bf d3 5a 26 6f 33 89 60 3b 9f a2 79 85 13 58 59 02 c1 2e d4 21 e5 e0 5e 01 2d ee a0 9b 39 e9 be 76 e8 4a 50 06 23 04 a3 4f 5e 1d 38 31 db c9 28 95 87 49 f6

The 4 equivalent subkeys:

K18=49 b1 1d d9 81 d6 48 05	K17=16 2f cc 02 61 ab d4 60
K16=61 44 ec c5 7e af b1 e5	K15=b6 48 63 a1 4e e9 89 ef

The two derived keys and the user key are：

K_{A1}= 07 4d 63 c2 6c f1 bd 8f 3d 3a 1e 75 a2 3f f4 f1	K_{L1}= a5 3e dd 22 0a ff 67 93 3c fe 60 6e fc fb 91 d0
K_{A2}= f8 b2 9c 3d 93 0e 42 70 c2 c5 e1 8a 5d c0 0b 0e	K_{L2}= 5a c1 22 dd f5 00 98 6c c3 01 9f 91 03 04 6e 2f

After key verifying, the correct initial Camellia key is: *K* = a5 3e dd 22 0a ff 67 93 3c fe 60 6e fc fb 91 d0

References:

[1] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. CRYPTO, volume 1109 of Lecture Notes in Computer Science, pages: 104–113, Springer. (1996)

[2] Boneh, D., DeMillo, R.A., Lipton, R.J. On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg. (1997)

[3] P. Kocher, J.Jaffe, B. Jun. Differential power analysis[A]. Proc. of Advances in Cryptology - CRYPTO '99 (M. Wiener, ed.), Springer-Verlag, 1999, LNCS 1666: 388-397. (1999)

[4] J.J. Quisquater, D. Samyde. Electromagnetic analysis (EMA): measures and countermeasures for smart cards, Smart cards programming and security (e-Smart 2001), Lectures Notes in Computer Science, vol. 2140: 200-210. Springer. (2001)

[5] Shamir, A. and Tromer, E. (2004). Acoustic cryptanalysis: On nosy people and noisy machines. Rump session of EuroCrypt 2004. http://www.wisdom.weizmann.ac.il/~tromer/acoustic/. (2004)

[6] Biham, E., Shamir, A. Differential fault analysis of secret key cryptosystem. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg. (1997)

[7] Biehl, I., Meyer, B., Muller, V. Differential fault analysis on elliptic curve cryptosystems. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 131–146. Springer, Heidelberg. (2000)

[8] Hemme, L.: A differential fault attack against early rounds of (Triple-) DES. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 254–267. Springer, Heidelberg. (2004)

[9] Blomer, J., Seifert, J.P.: Fault based cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg. (2003)

[10] Giraud, C.: DFA on AES. Cryptology ePrint Archive, http://eprint.iacr.org/2003/008. (2003)

[11] Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on AES. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg. (2003)

[12] Piret, G., Quisquater, J.J. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg. (2003)

[13] Debdeep Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In: B. Preneel (eds.) AFRICACRYPT 2009, LNCS 5580, pp. 421–434. (2009)

[14] Takahashi, J., Fukunaga, T., Yamakoshi, K. DFA mechanism on the AES schedule. In: Proceedings of 4th International Workshop on Fault Detection and Tolerance in Cryptography, FDTC, IEEE Computer Society, pp. 62–72. (2007)

[15] Takahashi, J., Fukunaga, T. Differential Fault Analysis on the AES Key Schedule, Cryptology ePrint Archive, http://eprint.iacr.org/2007/480. (2007)

[16] ZHOU Yongbin, WU Wengling, XU Nannan, FENG Dengguo. Differential Fault Attack on Camellia. Chinese Journal of Electronics, Vol.18, No.1, pp. 13–19. (2009)

[17] LI Wei, GU Dawu, LI Juanru . Differential fault analysis on the ARIA algorithm. Information Sciences. Elsevier Inc. pp.3727－3737. (2008)

[18] CHEN Hua, WU Wenling, and FENG Dengguo. Differential Fault Analysis on CLEFIA. In S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 284–295. Springer Heidelberg. (2007)

[19] Junko Takahashi and ToshinoriFukunaga. Improved Differential Fault Analysis on CLEFIA. Proceedings of the 2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC, IEEE Computer Society, pp 25-34. (2008)

[20] ZHANG Lei, WU Wenling, "Differential Fault Analysis on SMS4", Chinese Journal of Computers, Vol.29, No.9, pp. 1596–1602. (2006)

[21] LI Wi, GU Dawu. An improved method of differential fault analysis on the SMS4 cryptosystem[A]. The First International Symposium on Data, Privacy, and E-Commerce-ISDPE 2007[C]. Chengdu, China, IEEE Computer Society, pp.175-180. (2007)

[22] LI Wi, GU Dawu. Differential fault analysis on the SMS4 cipher by inducing faults to the key schedule. Journal on Communications. Vol.29, No.10, pp. 135–142. (2008)

[23] Hoch, J.J., Shamir, A.: Fault analysis of stream ciphers. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg. (2004)

[24] Biham, E., Granboulan, L., Nguyn, P.Q.: Impossible fault analysis of RC4 and differential fault analysis of RC4. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 359–367. Springer, Heidelberg. (2005)

[25] M. Hojsik and B. Rudolf. "Floating fault analysis of Trivium," In: D.R. Chowdhury, V. Rijmen, and A. Das (eds.) INDOCRYPT 2008. LNCS, Heidelberg,Springer,2008,vol. 5365, pp. 239–250. (2008)

[26] HU Yupu, GAO Juntao and Liu Qing. Hard Fault Analysis of Trivium. Cryptology ePrint Archive, http://eprint.iacr.org/2009/333. (2009)

[27] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita, Camellia: a 128-bit block cipher suitable for multiple platforms design and analysis, Proceedings of SAC '00, Lecture Notes in Computer Science 2012, pages: 39-56, Springer-Verlag. (2001)

[28] K. Aoki, T. Ichikawa, M. Kansa, M. Matsui, S. Moriai, Nakajima, and T. Tokita, "Specification of Camellia - a 128-bit Block Cipher", http://www.cosic.esat.kuleuven.be/nessie/workshop/submissions. (2000)

[29] D. Eastlake, "Additional XML Security Uniform Resource Indentifiers (URIs)", RFC4051. (2005)

[30] S. Moriai, A. Kato, M. Kanda, "Addition of Camellia Cipher Suites to Transport Layer Security (TLS)", RFC4132. (2005)

[31] A. Kato, S. Moriai, M.Kanda, "The Camellia Cipher Algorithm and Its Use With IPsec", RFC4312. (2005)

[32] OpenSSL the open-source toolkit for SSL / TLS [EB/OL], 2005. Available online at http://www.openssl.org/.