


ORIGINAL RESEARCH PAPER

Fault attacks on authenticated encryption modes for GIFT

Shuai Liu  | Jie Guan | Bin Hu

Department of Applied Mathematics, Strategic Support Force Information Engineering University, Zhengzhou, Henan, China

Correspondence

Shuai Liu, Department of Applied Mathematics, Strategic Support Force Information Engineering University, Zhengzhou, Henan, China.
Email: 379185570@qq.com

Funding information

National Natural Science Foundation of China, Grant/Award Numbers: 61272041, 61272488, 61572516

Abstract

There are several authenticated encryption modes for block cipher GIFT in the NIST lightweight cryptography standardisation process. In this study, the authors research on the fault attacks on this kind of authenticated encryption modes and mainly complete two tasks. First, the fault attack on the nonce-based authenticated encryption mode LOTUS/LOCUS is presented. At Asiacrypt2016, Dobraunig et al. showed the first fault attacks on several nonce-based authenticated encryption modes. Because LOTUS/LOCUS adopts the structure similar to XEX with secret nonce-dependent masks, their work is not applicable to LOTUS/LOCUS. A new fault attack is launched on LOTUS/LOCUS assuming that two bits can be made to reset in the fixed location during the encryption process. In this attack, neither plaintext nor ciphertext of the underlying block cipher is necessary to be known. To recover the correct key, a few hundred faulty ciphertexts are needed when transient faults are injected, while just one faulty ciphertext is sufficient for a permanent fault. Second, the Collision Fault Attack on GIFT is shown, in which 64 faulty ciphertexts are needed to recover the correct key. Based on this attack, authenticated encryption modes ESTATE_TweGIFT-128, GIFT-COFB and SUNDABE-GIFT are analysed and their keys are efficiently obtained with chosen nonce.

KEYWORDS

cryptography, message authentication hemorrhage, private key cryptography

1 | INTRODUCTION

Generally speaking, hardware devices can perform cryptographic operations correctly. However, in the case of external interference, hardware failures or operational errors may occur in the operation process of cryptographic modules. Fault attack uses these faulty behaviours or faulty messages to recover keys. Fault attacks mainly apply to the hardware implementation of cryptographic algorithms such as smartcards, IoT devices, etc. Due to its simplicity and effectivity, fault attack has become the one of the most popular side channel attack methods. The most common techniques to inject faults include variations in supply voltage [1], variations in the external clock [2], and laser fault injection [3].

Fault attack was first proposed by Boneh et al. [4] in 1996. They attacked the public key algorithm RSA using a computation error in the encryption process. Subsequently, researchers proposed different fault attack techniques according to the characteristics of various cryptosystems.

In 1997, Biham and Shamir [5] proposed the Differential Fault Attack (DFA) and attacked DES. DFA works by collecting correct and faulty ciphertexts corresponding to the same plaintext. The difference induced by the fault and the knowledge of ciphertexts can help to recover secret information. In 2000, Yen and Joye [6] showed the Safe Error Attack (SEA) on RSA. In 2006, Blömer and Krummel [7] proposed the Collision Fault Attack (CFA) against AES. Here, the attacker encrypts related plaintexts and injects a fault into the encryption process of one plaintext to get a collision. In 2013, Fuhr et al. [8] proposed the Statistical Fault Attack (SFA) for AES. One of the necessary conditions of SFA is that the output of cipher algorithm needs to be known. In recent years, various practical means of fault attacks have been proposed [9, 10], which also promote the emergence of methods and cryptosystems to resist fault attacks [11, 12].

While fault attacks on block cipher and stream cipher are widely concerned, there are few research studies on the fault attacks of authentication encryption schemes. An authentication

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2021 The Authors. *IET Information Security* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

encryption scheme, which guarantees both confidentiality and integrity, usually includes an encryption algorithm and a decryption algorithm. The encryption algorithm \mathcal{E} takes as input key K , nonce N , associated data A , and plaintext M , and outputs ciphertext C , and authentication tag T :

$$(C, T) = \mathcal{E}(K, N, A, M)$$

The decryption algorithm takes as input key K , nonce N , associated data A , ciphertext C , authentication tag T , and outputs plaintext M if the verification is successful or \perp if not:

$$\mathcal{D}(K, N, A, C, T) \in \{M, \perp\}$$

It is usually required that nonce is never repeated under the same key for encryption algorithm. We refer to such schemes as nonce-based authenticated encryption schemes. Even though some schemes claim security in nonce-misuse settings (such as repeated nonces, or release of unverified plaintext), this often leads to security degradation.

The unique nonce brings the unexpected result, which makes nonce-based authentication encryption schemes resist some fault attacks. In particular, DFA is rendered almost impossible, because DFA requires that the attacker gets both the correct and the faulty output corresponding to the same input, which is avoided in the nonce-based setting.

Early fault attacks on authenticated encryption schemes were proposed in nonce-misuse settings [13, 14]. In 2016, Dobraunig et al. [15] showed the first practical fault attack on several nonce-based authenticated encryption modes for AES, including the ISO/IEC standards GCM, CCM, EAX and OCB, as well as several second-round candidates of CAESAR competition.

At CHES 2016, Saha et al. [16] proposed a crafted fault attack technique, called internal differential fault analysis, to overcome the nonce barrier of DFA. Based on this, they attacked full-round PAEQ successfully. Their works [15, 16] are significant and inspire our work.

In 2018, NIST initiated a process to solicit and standardise lightweight cryptographic algorithms (hereafter, referred to as NIST-LWC) suitable for constrained environments. It is required that the submitted cryptographic algorithms are equipped with both authentication and encryption functions. In the second round of NIST-LWC, there are several authenticated encryption modes for block cipher GIFT or its tweakable variants, including LOTUS/LOCUS [17], ESTATE [18], GIFT-COFB [19], and SUNDAE-GIFT [20]. Especially, GIFT-COFB has been announced as one of the finalists. There are still no relevant results about fault attacks on these modes and in this study, we make an effort forward in this work.

1.1 | Our contribution

First, we propose the fault attack on the nonce-based authenticated encryption mode LOTUS/LOCUS. Because LOTUS/LOCUS adopts the structure similar to XEX with

secret nonce-dependent masks, previous fault attack techniques are not applicable to LOTUS/LOCUS. We give a new fault attack on LOTUS/LOCUS. In this attack, we assume that we can reset two bits in the fixed location during the encryption process. Then, we get a sample sequence of a discrete random variable whose distribution is determined by single-bit secret key information. By distinguishing two discrete random variables, we can recover this single-bit key information. In the whole attack process, neither plaintext nor ciphertext of underlying block cipher is necessary to be known. To recover the complete key, a few of faulty ciphertexts are needed when transient faults are injected, while just one faulty ciphertext is sufficient when a permanent fault is injected.

Second, we show the Collision Fault Attack on GIFT, where 64 faulty ciphertexts are needed to recover the correct key. Based on this attack, we analyse three authenticated encryption modes ESTATE_TweGIFT-128, GIFT-COFB and SUNDAE-GIFT, and efficiently recover their keys in the nonce-chosen setting.

1.2 | Outline

This study is organized as follows. In Section 2, we introduce some background on GIFT algorithm and its tweakable variants, as well as other related information. In Section 3, we propose the differential statistical fault attack on LOTUS/LOCUS. In Section 4, we launch the Collision Fault Attack on GIFT algorithm, then apply the same measure to analyse the authenticated encryption modes ESTATE_TweGIFT-128, GIFT-COFB and SUNDAE-GIFT.

2 | BACKGROUND

In this section, we start with a general overview of GIFT algorithms, including GIFT-64 and GIFT-128. Then, we revisit the multiplication of any field element with two in the finite field $\mathbb{F}_{2^{128}}$. Finally, we introduce the optimal distinguisher between two discrete random variables. As for the authenticated encryption modes involved in this study, we will give a brief description when necessary, and we direct the readers to refer [17–20] for more details.

2.1 | GIFT algorithm

At CHES2017, Banik et al. [21] proposed the GIFT algorithm, which is a lightweight block cipher algorithm with the SPN structure, including GIFT-64-128 and GIFT-128-128. GIFT-64-128 is a 28-round cipher with a block length of 64-bit and a key length of 128-bit. GIFT-128-128 is a 40-round cipher with a block length of 128-bit and a key length of 128-bit. We call them GIFT-64 and GIFT-128, respectively. Figure 1 illustrates 2 rounds of GIFT-64.

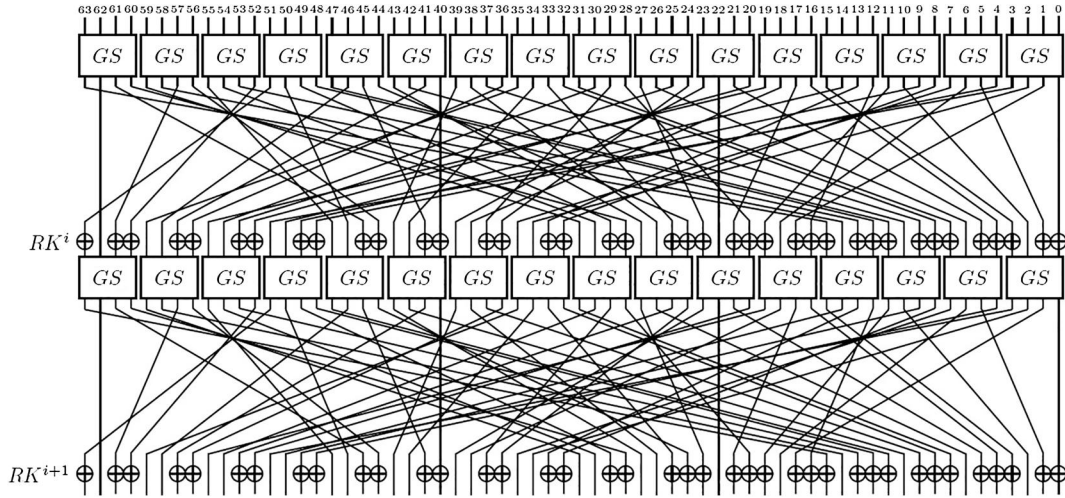


FIGURE 1 2 rounds of GIFT-64

The state S of GIFT can be expressed as an n -bit string $b_{n-1}b_{n-2}\dots b_0$, where $n = 64, 128$ and b_0 is the least significant bit. It also can be expressed as s many 4-bit nibbles $S = S[s-1]||S[s-2]||\dots||S[0]$ ($s = 16, 32$). A 128-bit key K is expressed as $K = k_7||k_6||\dots||k_0$, where k_i is a 16-bit word. Each round of GIFT includes 4 steps: SubCells, PermBits, AddRoundKey, and AddRoundConstant (AddRoundConstant is omitted in Figure 1).

2.1.1 | Sub cells

Both versions of GIFT use the same 4-bit Sbox GS , which is applied to every nibble of the cipher state

$$S[i] \leftarrow GS(S[i]) \quad i \in \{0, \dots, s-1\}$$

The Sbox is given in Table 1.

2.1.2 | Perm bits

It maps bits from bit position i of the cipher state to bit position $P(i)$

$$b_{P(i)} \leftarrow b_i, \quad i \in \{0, \dots, n-1\}$$

The permutations of GIFT-64 and GIFT-128 can be, respectively, expressed as follows:

$$P_{64}(i) = 4 \left\lfloor \frac{i}{16} \right\rfloor + 16 \left(\left(3 \left\lfloor \frac{i \bmod 16}{4} \right\rfloor + (i \bmod 4) \right) \bmod 4 \right) + (i \bmod 4)$$

$$P_{128}(i) = 4 \left\lfloor \frac{i}{16} \right\rfloor + 32 \left(\left(3 \left\lfloor \frac{i \bmod 16}{4} \right\rfloor + (i \bmod 4) \right) \bmod 4 \right) + (i \bmod 4)$$

TABLE 1 Sbox of GIFT

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$GS(x)$	1	a	4	c	6	f	3	9	2	d	b	7	5	0	8	e

2.1.3 | Key schedule

Both versions of GIFT use the same key schedule. A round key $RK = U||V$ is extracted from the key state before the key state is updated. For GIFT-64, $U \leftarrow k_1$, $V \leftarrow k_0$. For GIFT-128, $U \leftarrow k_5||k_4$, $V \leftarrow k_1||k_0$.

The key state is updated as follows:

$$k_7||k_6||\dots||k_1||k_0 \leftarrow k_1 >>> 2||k_0 >>> 12||\dots||k_3||k_2$$

2.1.4 | Add round key

The $n/2$ -bit round key is expressed as $RK = U||V = u_{s-1}\dots u_0||v_{s-1}\dots v_0$, where $s = 16, 32$ for GIFT-64 and GIFT-128, respectively.

For GIFT-64, U and V are XORed to $\{b_{4i+1}\}$ and $\{b_{4i}\}$ of the cipher state, respectively.

$$b_{4i+1} \leftarrow b_{4i+1} \oplus u_i, \quad b_{4i} \leftarrow b_{4i} \oplus v_i, \quad i \in \{0, \dots, 15\}$$

For GIFT-128, U and V are XORed to $\{b_{4i+2}\}$ and $\{b_{4i+1}\}$ of the cipher state, respectively.

$$b_{4i+2} \leftarrow b_{4i+2} \oplus u_i, \quad b_{4i+1} \leftarrow b_{4i+1} \oplus v_i, \quad i \in \{0, \dots, 31\}$$

2.1.5 | Add round constant

For both versions of GIFT, a single bit '1' and a 6-bit round constant $C = c_5c_4c_3c_2c_1c_0$ are XORed to the cipher state at the bit position $n-1, 23, 19, 15, 11, 7$ and 3, respectively.

$$\begin{aligned}
b_{n-1} &\leftarrow b_{n-1} \oplus 1 & b_{23} &\leftarrow b_{23} \oplus c_5 & b_{19} &\leftarrow b_{19} \oplus c_4 \\
b_{15} &\leftarrow b_{15} \oplus c_3 & b_{11} &\leftarrow b_{11} \oplus c_2 & b_7 &\leftarrow b_7 \oplus c_1 \\
b_3 &\leftarrow b_3 \oplus c_0
\end{aligned}$$

The detail about round constant generation can be found in [21]. Table 2 gives the values of the constants for each round.

2.2 | Finite field arithmetic

Finite field $\mathbb{F}_{2^{128}}$ can be represented by the primitive polynomial

$$P(x) = x^{128} + x^7 + x^2 + x + 1$$

The primitive element of $\mathbb{F}_{2^{128}}$ is 2. The multiplication of any field element with 2 can be computed efficiently as follows:

$$\beta \odot 2 = \begin{cases} \beta \ll 1 & \text{if } \beta_{127} = 0 \\ (\beta \ll 1) \oplus 0^{120}10000111 & \text{if } \beta_{127} = 1 \end{cases} \quad (1)$$

2.3 | Optimal distinguisher between two discrete random variables

In the fault attack on LOTUS/LOCUS, the problem on how to distinguish two discrete random variables will be involved. Here, we revisit the result about the optimal distinguisher [22], as well as the success probability in Theorem 1.

Theorem 1 [22] Assume that ξ_0 and ξ_1 are two discrete random variables. Table 3 gives their distributions, where $\frac{1}{2} < \bar{p} \leq 1$. Let $\lambda \in \{\xi_0, \xi_1\}$ with $P(\lambda = \xi_0) = P(\lambda = \xi_1) = \frac{1}{2}$. Given the sample sequence $a = (a_1, a_2, \dots, a_n)$ of λ with length n , the optimal distinguisher to determine whether $\lambda = \xi_0$ or $\lambda = \xi_1$ is

$$\lambda = \begin{cases} \xi_0 & n_0 \geq n_1 \\ \xi_1 & n_0 < n_1 \end{cases}$$

TABLE 2 Round constant of GIFT

Rounds	Constants
1-16	01,03,07,0F,1 F,3E,3D,3B,37,2F,1 E,3C,39,33,27,0E
17-32	1D,3 A,35,2B,16,2C,18,30,21,02,05,0B,17,2E,1 C,38
33-48	31,23,06,0D,1 B,36,2D,1 A,34,29,12,24,08,11,22,04

TABLE 3 Probability distribution table of discrete random variables ξ_0 and ξ_1

x	0	1	x	0	1
$P(\xi_0 = x)$	\bar{p}	$1 - \bar{p}$	$P(\xi_1 = x)$	$1 - \bar{p}$	\bar{p}

where n_0 and n_1 represent the number of 0 and one in sequence a , respectively. The success probability of the decision is

$$\begin{aligned}
p_{\text{decision_success}} &= \frac{1}{2} \left(\sum_{n_0=\lfloor \frac{n}{2} \rfloor}^n C_n^{n_0} \bar{p}^{n_0} (1 - \bar{p})^{n-n_0} \right. \\
&\quad \left. + \sum_{n_0=0}^{\lfloor \frac{n}{2} \rfloor - 1} C_n^{n_0} (1 - \bar{p})^{n_0} \bar{p}^{n-n_0} \right)
\end{aligned}$$

3 | DIFFERENCE-STATISTICAL FAULT ATTACK ON LOTUS/LOCUS

Because LOTUS/LOCUS adopts the structure similar to XEX with secret nonce-dependent masks, it is impossible for attackers to know or control the input and output of the underlying block cipher. As a result, previous fault attack techniques are not applicable to LOTUS/LOCUS in the nonce-based setting. In 2017, Mahri et al. [23] researched on the fault attack on the similar structure, but with a strong assumption of the fault model. In this section, we launch a new fault attack to recover the correct key of LOTUS/LOCUS, which is called the difference-statistical fault attack. For simplicity, we have only provided the attack process on LOTUS, all work in this section is completely applicable to LOCUS.

3.1 | Structure analysis of LOTUS and TweGIFT-64

Tweakable block cipher TweGIFT-64 is the underlying cryptographic module of LOTUS [17], denoted by $\bar{E}_{x,y}$, where x and y represent key and tweak, respectively. TweGIFT-64 is a tweakable variant of GIFT-64 with a 4-bit tweak, which is expanded and then XORed to cipher state of round 3, 7, 11, 15, 19 and 23. Except for that, TweGIFT-64 is completely same with GIFT-64.

The encryption algorithm of LOTUS takes as input the 128-bit secret key $K = (K_{127}, K_{126}, \dots, K_0)$, 128-bit public nonce $N = (N_{127}, N_{126}, \dots, N_0)$, associated data A and plaintext M of arbitrary length. The entire encryption process consists of three phases: initialisation phase, associated data processing phase, and plaintext processing phase. Hereinafter, we refer to the bit with subscript i when we say i th bit of a state vector.

3.1.1 | Initialisation

Generate secret parameters

$$K_N = K \oplus N$$

$$\Delta_N = \bar{E}_{K_N,1}(\bar{E}_{K,0}(0^n))$$

3.1.2 | Associated date processing

Parse the date into 64-bit blocks $A \rightarrow A_0 A_1 \dots A_{l_A-1}$ and process them in a similar manner as the hash layer of PMAC [24], where l_A denotes the number of associated date blocks.

3.1.3 | Plaintext processing

Parse the plaintext into 64-bit blocks $M \rightarrow M_0 M_1 \dots M_{l_M-1}$ and process them in a similar way as OTR [25]. Regard (M_{2i}, M_{2i+1}) as a plaintext di-block. For each plaintext di-block, a simple variant of two-round Feistel is applied and it outputs a ciphertext di-block (C_{2i}, C_{2i+1}) (the detail is showed in the left half of Figure 2, where $L_i = 2^{i+1+l_A} K_N$). The output of the entire plaintext processing phase is the ciphertext $C = (C_0 || C_1 || \dots || C_{l_M-1})$.

From the above description, Δ_N is unknown and changes in every encryption along with nonce. All previous fault attack techniques require the control on the input or output of the underlying cipher. However, in the whole encryption process of LOTUS, the input and output of TweGIFT-64 is either the intermediate value, which is not outputted or XORed with Δ_N . That is, the input and output of TweGIFT-64 are unknown and random. Even though [15] has given the SFA on XEX-like constructions, it requires that X is fixed and related only to the key, which is obviously not satisfied by the nonce-dependent Δ_N . Then, we will show how to overcome the influence of Δ_N .

Figure 2 gives the processing of two consecutive plaintext di-blocks (M_{2i}, M_{2i+1}) and (M_{2i+2}, M_{2i+3}) , which is called a *targetted structure* with serial number i . Let BC_i and BC_{i+1} represent the output of $\bar{E}_{L_i,5}(W_{2i+1})$ and $\bar{E}_{L_{i+1},5}(W_{2i+3})$ (marked with dotted box in Figure 2), respectively. z_i and z_{i+1} represent the 53-rd bit in the cipher state after the last-round SubCells of $\bar{E}_{L_i,5}(W_{2i+1})$ and $\bar{E}_{L_{i+1},5}(W_{2i+3})$, respectively. In a *targetted structure*, we can easily get

$$\begin{aligned} (M_{2i} \oplus C_{2i+1}) \oplus (M_{2i+2} \oplus C_{2i+3}) \\ = (\Delta_N \oplus BC_i) \oplus (\Delta_N \oplus BC_{i+1}) \\ = BC_i \oplus BC_{i+1} \end{aligned} \quad (2)$$

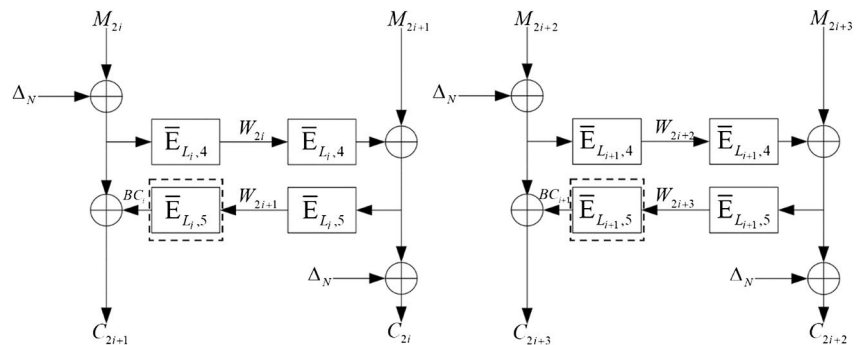


FIGURE 2 Processing of two consecutive plaintext di-blocks in LOTUS

It is easy to see that we can get the output difference of $\bar{E}_{L_i,5}(W_{2i+1})$ and $\bar{E}_{L_{i+1},5}(W_{2i+3})$ with the knowledge of the plaintext M and ciphertext C .

In the associated date processing and plaintext processing phase, the key of TweGIFT-64 $L = K_N$ is updated by 2-multiplication before the process of a block or a di-block. From section 2.2, it is easy to see that the most left bits will be removed after 2-multiplication, which makes that bit cannot be recovered by our way. Therefore, we make the associated date empty to reduce the time of 2-multiplication and guarantee that we can recover as many key bits as possible. At least 2-multiplication occurs before L is used as the key of TweGIFT-64. Because the most right seven bits $K_6 \oplus N_6, \dots, K_0 \oplus N_0$ may be changed after 2-multiplication, those bits make the situation more complicated and thus we will ignore them. Property 1 gives a detailed analysis on how key bits participate in the encryption process.

Property 1 For LOTUS, suppose the secret key K is fixed, nonce N is uniformly random, associated date A is empty, and plaintext M is long enough, we have the following: for $0 \leq i \leq 119$, the bit z_i in the last round of $\bar{E}_{L_i,5}$ is XORed to $K_{126-i} \oplus N_{126-i}$ to get the 13-th bit of BC_i .

Proof. The initialisation phase outputs $K_N = K \oplus N$. Since the associated date is empty, we have $l_A = 0$. In the processing of i th ($i \geq 0$) plaintext di-block, the secret key of $\bar{E}_{L_i,5}$ is $L_i = 2^{i+1}(K \oplus N)$. According to the expression (1), the left $120 - i$ bits of L_i ($0 \leq i \leq 119$) are as follows:

$$K_{126-i} \oplus N_{126-i}, K_{125-i} \oplus N_{125-i}, \dots, K_7 \oplus N_7.$$

Observing the key schedule of TweGIFT-64 (same with GIFT-64 showed in Section 2), we can find that the above property is obvious.

Since a secret nonce-dependent mask Δ_N is used in the processing of each plaintext di-block, the input and output of the underlying block cipher are unknown and uncontrolled. In the nonce-based setting, previous fault attack techniques, such as DFA, CFA and SFA, are not applicable to LOTUS. We launch a new fault attack on LOTUS in the remainder of this section, which can recover the correct key of LOTUS efficiently.

3.2 | Single-bit key information recovery

We can recover single-bit key information, the difference of two key bits, by executing the attack once as shown in the following. Assume that the secret key K is fixed, nonce N is chosen uniformly at random, associated data A is empty, and the plaintext M is long enough.

3.2.1 | Fault model

In an encryption process of LOTUS, inject faults into the bits z_i and z_{i+1} in i th *targetted structure* to make the two bits to be reset (reset means that the value of the bit becomes 0). Then, get the faulty ciphertext \tilde{C} after every encryption process. The two faulty bits are denoted by \tilde{z}_i and \tilde{z}_{i+1} .

We will show how to recover single-bit key information. Let sum represent the difference of the two faulty bits, that is

$$sum = \tilde{z}_i \oplus \tilde{z}_{i+1}$$

Here, it is not required that fault injection always succeeds in every encryption process. It is sufficient to make the values of sum tend to 0 in repeated trials. Assume that the success probability of fault injection is p ($0 < p \leq 1$) in every encryption process. If fault injection succeeds, we have $sum = 0$. If not, sum is uniformly distributed in $\{0, 1\}$. We can get the distribution of discrete random variable sum :

$$P(sum = 0) = p + \frac{1}{2}(1 - p) = \frac{1}{2} + \frac{1}{2}p$$

$$P(sum = 1) = \frac{1}{2}(1 - p) = \frac{1}{2} - \frac{1}{2}p$$

Property 1 shows the fact that the two faulty bits are XORed to $K_{126-i} \oplus N_{126-i}$ and $K_{125-i} \oplus N_{125-i}$, respectively, to get the 13-th bit of BC_i and BC_{i+1} . From Equation (2), we have the following:

$$sum \oplus (K_{126-i} \oplus N_{126-i}) \oplus (K_{125-i} \oplus N_{125-i}) = (BC_i \oplus BC_{i+1})_{[13]} = ((M_{2i} \oplus \tilde{C}_{2i+1}) \oplus (M_{2i+2} \oplus \tilde{C}_{2i+3}))_{[13]}$$

where the subscript $_{[13]}$ represents the 13-th bit of a state vector. Simplify the above equation:

$$sum \oplus K_{126-i} \oplus K_{125-i} = \left((M_{2i} \oplus \tilde{C}_{2i+1}) \oplus (M_{2i+2} \oplus \tilde{C}_{2i+3}) \right)_{[13]} \oplus N_{126-i} \oplus N_{125-i} \quad (3)$$

Define three discrete random variables $\lambda = sum \oplus K_{126-i} \oplus K_{125-i}$, $\xi_0 = sum$, and $\xi_1 = sum \oplus 1$. If $K_{126-i} \oplus K_{125-i} = 0$, we have $\lambda = \xi_0$. If $K_{126-i} \oplus K_{125-i} = 1$, $\lambda = \xi_1$. So,

$$P(\lambda = \xi_0) = P(K_{126-i} \oplus K_{125-i} = 0) = \frac{1}{2}$$

$$P(\lambda = \xi_1) = P(K_{126-i} \oplus K_{125-i} = 1) = \frac{1}{2}$$

According to Equation (3), we can get an n -length sample sequence $a = (a_1, a_2, \dots, a_n)$ of λ by injecting the above faults in n encryption processes and getting n faulty ciphertexts. Then, we decide whether $\lambda = \xi_0$ or $\lambda = \xi_1$ according to Theorem 1 (n_0 and n_1 represent the number of 0 and the number of 1 in the sequence a , respectively):

$$\lambda = \begin{cases} \xi_0 & n_0 \geq n_1 \\ \xi_1 & n_0 < n_1 \end{cases}$$

We will get $K_{126-i} \oplus K_{125-i} = 0$ if $\lambda = \xi_0$, and $K_{126-i} \oplus K_{125-i} = 1$ if $\lambda = \xi_1$. According to Theorem 1, the success probability of recovering single-bit key information is

$$p_{\text{single_bit}} = \frac{1}{2^{n+1}} \left(\sum_{n_0=\lfloor \frac{n}{2} \rfloor}^n C_n^{n_0} (1+p)^{n_0} (1-p)^{n-n_0} + \sum_{n_0=0}^{\lfloor \frac{n}{2} \rfloor - 1} C_n^{n_0} (1-p)^{n_0} (1+p)^{n-n_0} \right)$$

TABLE 4 The success probability of recovering single-bit key information

Success pr. of fault injection (p)	Nb. of faulty ciphertexts (n)	Success pr. ($p_{\text{single_bit}}$)
0.8	1	0.90
	3	0.97
	5	0.99
0.7	3	0.94
	5	0.97
	9	0.99
0.6	5	0.94
	7	0.97
	13	0.99
0.5	7	0.93
	9	0.95
	19	0.99
0.4	9	0.90
	17	0.96
	31	0.99

For simplicity, we denote this expression by $p_{\text{single_bit}} = f(p, n)$. The factors affecting the success probability $p_{\text{single_bit}}$ include the success probability of fault injection p and the number of faulty ciphertexts n . Table 4 shows the theoretical value of $p_{\text{single_bit}}$ corresponding to different p and n .

3.3 | Complete key recovery

The complete key K of LOTUS can be recovered by using the approach in Section 3.2. The Algorithm 1 shows the entire attack process. $\text{LOTUS_}\tilde{\text{E}}(K, i)$ denotes the encryption algorithm of LOTUS with faults injected into the bits z_i and z_{i+1} of i th *targetted structure*, in which the nonce and plaintexts are chosen randomly. $\#$ denotes the number of elements in a set.

Algorithm 1 The key recovery attack on LOTUS

Input: The number of bits recovered by fault attack: m ($1 \leq m \leq 119$), the number of faulty ciphertexts used to recover single-bit key information: n .

Output: The correct key K .

Step1:

```

for  $i = 0$  to  $m - 1$  do
  for  $j = 0$  to  $n - 1$  do
     $\tilde{C} = \text{LOTUS\_}\tilde{\text{E}}(K, i)$ 
     $a_j = ((M_{2i} \oplus \tilde{C}_{2i+1}) \oplus (M_{2i+2} \oplus \tilde{C}_{2i+3}))_{[13]} \oplus$ 
       $N_{126-i} \oplus N_{125-i}$ 
  end for
   $a = (a_0, a_1, \dots, a_{n-1})$ 
   $n_0 = \#\{a_j = 0 \mid j \in \{0, 1, \dots, n-1\}\}$ 
   $n_1 = \#\{a_j = 1 \mid j \in \{0, 1, \dots, n-1\}\}$ 
  If  $n_0 \geq n_1$  do
     $K_{126-i} \oplus K_{125-i} = 0$ 
  else do
     $K_{126-i} \oplus K_{125-i} = 1$ 
  end if
end for

```

Step2: Guess remainder key bits K_{127} and $\{K_{126-m}, \dots, K_0\}$. If the correct key is not recovered, jump to Step1.

In Algorithm 1, after Step 1, we get the equation set

$$\begin{cases} K_{126} \oplus K_{125} = c_0 \\ K_{125} \oplus K_{124} = c_1 \\ \vdots \\ K_{128-m} \oplus K_{127-m} = c_{m-2} \\ K_{127-m} \oplus K_{126-m} = c_{m-1} \end{cases}$$

The probability of recovering the correct key by executing Step 1-Step 2 once is $p_s = f(p, n)^m$, that is the probability of the above equation set being correct.

3.3.1 | Complexity analysis

To recover the correct key K , the average times of executing Step 1-Step 2 is

$$\sum_{i=1}^{\infty} i \cdot p_s \cdot (1 - p_s)^{i-1} = p_s \sum_{i=1}^{\infty} i \cdot (1 - p_s)^{i-1} = \frac{1}{p_s}$$

As a result, the total number of faulty ciphertexts needed is $\frac{m \cdot n}{p_s}$ and the computation complexity is $\frac{2^{128-m}}{p_s}$ on average.

The factors affecting the total number of faulty ciphertexts includes parameters p , m , and n . The following conclusions are made:

- (1) The greater p is, the less is the total faulty ciphertexts.
- (2) The smaller m is, the less is the total faulty ciphertexts.

By fixing p and m , we can get the value of n to minimise the total number of faulty ciphertexts by solving linear programming problems. Table 5 gives the expense of recovering key corresponding to different m and p , where the value of n is chosen to minimise the total number of faulty ciphertexts.

3.4 | Fault injection

Chakraborti et al. [17] provided the round-based hardware implementation of TweGIFT-64 as the underlying module of LOTUS/LOCUS. Interestingly, the fault injection position in our attack is fixed at the 53-rd bit in the cipher state after the last-round SubCells of TweGIFT-64, which means that the faulty bits occur at the same location throughout the attack. This would provide convenience and flexibility for fault injection. Here, we recommend two feasible methods to inject faults, including optical fault injection and permanent fault injection. Optical fault is a kind of transient fault, which will not affect the subsequent operation of the hardware equipment.

TABLE 5 The expense of recovering key corresponding to different m and p

m	98			108		
p	0.9	0.8	0.7	0.9	0.8	0.7
n	5	7	11	5	7	11
Total number of Faulty ciphertexts	549	897	1399	612	1015	1583
Computation complexity	$2^{30.16}$	$2^{30.39}$	$2^{30.38}$	$2^{20.18}$	$2^{20.43}$	$2^{20.41}$

3.4.1 | Optical fault

In 2002, Sergei et al. [3] proposed a practical fault injection technique. They used a microscope and a flashgun to implement fault injection on secure microcontrollers and smartcards, and set or reset any individual bit at a low cost. In the previous attack described in Section 3.2 and Section 3.3, two such faults are injected at the same location but with different time points in one encryption process of LOTUS. Using the technique of Sergei et al., we just need to fix the location of the cryptographic hardware devices, microscope and light source in advance, and control the time to produce faults.

3.4.2 | Permanent fault

We can generate a permanent fault in the targetted bit, either by destroying the bit or by cutting the wire which enters or leaves this cell, and the value of the bit will be permanently fixed. Therefore, it is sufficient to encrypt just one plaintext (including at least 120 plaintext di-blocks with the empty associated data) and get the corresponding faulty ciphertext, and then the 119-bit key information can be recovered at one time. The value of sum in the i th ($0 \leq i \leq 118$) *targetted structure* is always 0. From Equation (3), we have

$$K_{126-i} \oplus K_{125-i} = \left((M_{2i} \oplus \tilde{C}_{2i+1}) \oplus (M_{2i+2} \oplus \tilde{C}_{2i+3}) \right)_{[13]} \oplus N_{126-i} \oplus N_{125-i}$$

Then, get the equation set

$$\begin{cases} K_{126} \oplus K_{125} = c_0 \\ K_{125} \oplus K_{124} = c_1 \\ \vdots \\ K_9 \oplus K_8 = c_{117} \\ K_8 \oplus K_7 = c_{118} \end{cases}$$

The equation set can uniquely determine 119-bit key information; by exhausting the remaining nine key bits, we can recover the correct key K .

3.5 | Experimental results

We implement computer simulations of Difference-Statistical Fault Attack on LOTUS under the assumption of the transient fault. The simulation is performed thousands of times over randomly chosen keys, nonces and plaintexts. The results for different values of parameters are shown in the form of a bar diagram in Figure 3.

In Figure 3, the abscissa axis and vertical axis express the value of parameter n and the total number of faulty ciphertexts. The six subplots are corresponding to different values of p and m . In the first subplot, under the condition of $p = 0.9$ and $m = 98$, the total number of faulty ciphertexts needed is minimal when $n = 5$. We can see that the experimental result is consistent with the theoretical value given in Table 5.

In practice, the value of m can be reduced further to 88, then the total number of faulty ciphertexts can be as low as 400

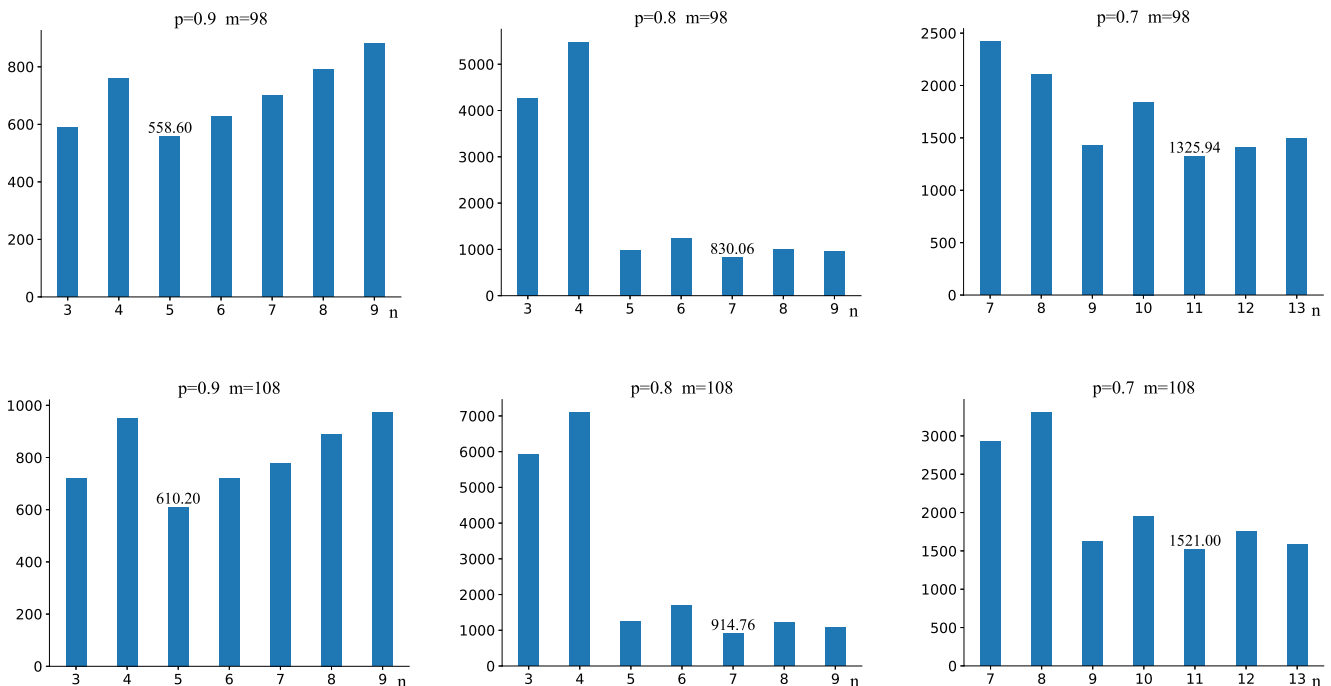


FIGURE 3 The experimental results of difference-statistical fault attack on LOTUS over different values of m and p

with a feasible computation complexity about 2^{40} (when $p = 0.9$ and $n = 3$).

3.5.1 | Summary

This section gives the Difference-Statistical Fault Attack on LOTUS, which can recover one-bit key information every time. Combined with the key schedule of LOTUS, up to 119-bit key information can be recovered. The complete key can be recovered by exhausting the remaining key bits. Since LOCUS has the similar structure to LOTUS, all the work in this section is applicable to LOCUS.

4 | COLLISION FAULT ATTACK ON THREE AUTHENTICATED ENCRYPTION MODES FOR GIFT

For ESTATE_TweGIFT-128, GIFT-COFB, and SUNDABE-GIFT, the situation is different without the application of Δ_N . Therefore, we considered whether the existing fault attack techniques are applicable. These three authenticated encryption modes are all based on GIFT-128 or its tweakable variant. Therefore, we first analyse the attack on GIFT-128.

Because of the existence of nonce, DFA is excluded, in which encrypting the same input twice is required. Then, we research the SFA on GIFT-128, and we find that a direct application of SFA on GIFT-128 is impracticable (the detailed analysis and experiments can be found in Appendices 8.1). The intrinsic reason is the simple round function of GIFT-128. The distribution deriving wrong keys is not random. Therefore, we choose collision fault attack, and show that CFA can also overcome the nonce barrier, which is not mentioned in the previous work.

In this section, we first launch the collision fault attack on GIFT-128. Based on this, we propose the collision fault attack on authenticated encryption modes ESTATE_TweGIFT-128, GIFT-COFB, and SUNDABE-GIFT assuming that nonce cannot be repeated but can be chosen. The fault type involved in this section is the bit flip (0 to 1, 1 to 0).

In the following, for a 128-bit cipher state, we can regard it as a 128-bit string and the serial number of each bit is $0, 1, \dots, 127$ from right to left. We can also regard it as 32 4-bit nibbles, and let $S[i]$ represent i th nibble. We denote the faulty state by $\tilde{\cdot}$.

4.1 | Collision fault attack on GIFT-128

From the key schedule of GIFT-128 described in Section 2.1, the round key of the first round is $U_1 = k_5 \| k_4$ and $V_1 = k_1 \| k_0$, the round key of the second round is $U_2 = k_7 \| k_6$ and $V_2 = k_3 \| k_2$. It is obvious that the correct key K can be

recovered with the knowledge of the round keys of the first two rounds.

Figure 4 shows the partial structure of the first two rounds of GIFT-128, where $GS^{r,i}$ represents the i th Sbox of r th round.

For $d \in \{0,1\}^4$, define $T[d] = \{x \in \{0,1\}^4 | GS(x) \oplus GS(x \oplus d) = 1000\}$. According to the differential characteristics of the GIFT Sbox, we have $|T[d]| \in \{0, 2\}$. Table 6 lists all possible cases of $T[d]$.

We define the plaintext set (r, i) -set = $\{P_0, P_1, \dots, P_{15}\}$ ($i \in \{0, 1, \dots, 31\}, r \in \{1, 2\}$) satisfying the following: after the r th round PermBits operation, the four bits of the cipher state with serial number $4i, 4i + 1, 4i + 2, 4i + 3$ traverses through all values in $\{0,1\}^4$, and other bits are identical.

We will show how to recover two key bits. Choose a $(1, 0)$ -set, let Y_j denote the output of $GS^{2,0}$ corresponding to P_j ($0 \leq j \leq 15$). Encrypt P_1, \dots, P_{15} and get the correct ciphertexts C_1, \dots, C_{15} . When encrypting P_0 , we inject a single-bit fault to make the most left bit of Y_0 flip and get the faulty ciphertext \tilde{C}_0 . Since Y_j traverses through $\{0,1\}^4$, there exists an integer $g \in \{1, \dots, 15\}$ such that $\tilde{Y}_0 = Y_g$, then we have $\tilde{C}_0 = C_g$.

For P_0 and P_g , the corresponding input difference of $GS^{2,0}$ is $\Delta = (PB(GS(P_0)) \oplus PB(GS(P_g)))[0]$ and the output difference is 1000. Let $X_0 = (PB(GS(P_0)))[0] \oplus 0K_{64}K_00 \oplus 1000$ be the input of $GS^{2,0}$ in the encryption process of P_0 , we have $X_0 \in T[\Delta]$.

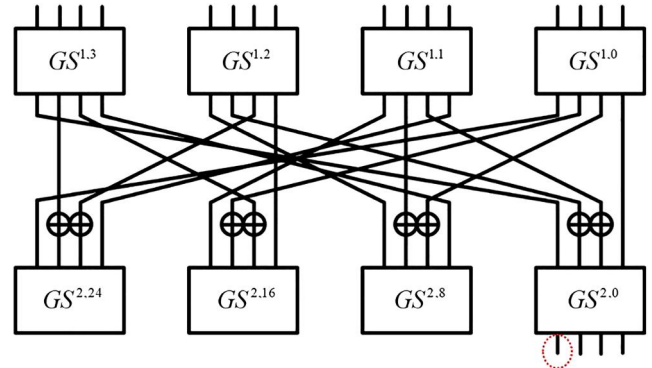


FIGURE 4 Illustration of the collision fault attack on GIFT-128

TABLE 6 All possible inputs of GIFT Sbox with the output difference 1000

Input difference	Possible inputs
0001	0010, 0011
0011	1101, 1110
0101	1001, 1100
0111	0000, 0111
1001	0001, 1000
1011	0100, 1111
1100	0110, 1010
1110	0101, 1011

Since the most left bit and the most right bit of X_0 can be uniquely determined, X_0 can be uniquely determined according to Table 6. Then, we can recover key bits K_{64} and K_0 .

Using the above way, the first-round key can be recovered, and based on this the second-round key is recovered. Algorithm 2 shows the complete process to recover the key K . $\text{GIFT-128_E}()$ denotes the encryption algorithm of GIFT-128. $\text{RoundFunction}_1()$ denotes the first-round round function of GIFT-128. $\text{GIFT-128_}\tilde{\text{E}}_{r,i}()$ denotes the encryption algorithm of GIFT-128 with the fault injected to make the most left bit of the output of $GS^{r+1,i}$ flip.

Algorithm 2 The Collision Fault Attack on GIFT-128

Output: The correct key K .

```

for  $r = 1$  to 2 do
  for  $i = 0$  to 31 do
    Choose a  $(r, i)$ -set =  $\{P_0, P_1, \dots, P_{15}\}$ 
    for  $j = 1$  to 15 do
       $C_j = \text{GIFT-128\_E}(P_j)$ 
    end for
     $\tilde{C}_0 = \text{GIFT-128\_}\tilde{\text{E}}_{r,i}(P_0)$ 
    Find  $g \in \{1, \dots, 15\}$  satisfying  $\tilde{C}_0 = C_g$ 
    if  $r = 1$  do
       $\Delta = (PB(GS(P_0)) \oplus PB(GS(P_g))) [i];$ 
       $X = (PB(GS(P_0))) [i] \oplus 0K_{64+i}K_i0 \oplus RC;$ 
    end if
    if  $r = 2$  do
       $S_0 = \text{RoundFunction}_1(P_0)$ 
       $S_1 = \text{RoundFunction}_1(P_g)$ 
       $\Delta = (PB(GS(S_0)) \oplus PB(GS(S_1))) [i];$ 
       $X = (PB(GS(S_0))) [i] \oplus 0K_{96+i}K_{32+i}0 \oplus RC;$ 
    end if
    According to the possible inputs of  $T[\Delta]$ 
    showed in Table 6, determine the value of
     $X$  and then recover key bits  $K_{32r+32+i}$  and
     $K_{32r-32+i}$ .
  end for
end for
Finally, output the complete key  $K = (K_{127},$ 
 $K_{126}, \dots, K_0)$ .

```

To recover the key of GIFT-128, just 64 faulty ciphertexts are needed. The Algorithm 2 is also applicable to TweGIFT-128.

At the end of this part, we will explain why we choose to inject fault into the most left bit of a nibble.

When the fault is injected into other three bits of the output of Sbox, output differences 0100, 0010, and 0001 are generated, respectively. From Table 7, it can be seen that the input of Sbox cannot always be determined uniquely by the known information. For example, when output difference is 0001 and input difference is 1010, there are two possible inputs though the 0-th and 3-th bit of input is known. That is, the key cannot be determined uniquely. As a result, more faulty ciphertexts will be needed to recover the key. To sum up, injecting fault into the most left bit of a nibble is the best choice.

4.2 | Collision fault attack on authenticated encryption modes

In this section, we give the collision fault attacks on three authenticated encryption modes for GIFT, including ESTATE_TweGIFT-128 (based on TweGIFT-128), GIFT-COFB, and SUNDAAE-GIFT (based on GIFT-128) assuming that nonce cannot be repeated but can be chosen.

First, let us recall the conditions of Section 4.1 that are necessary for the collision fault attack to work:

1. The block cipher inputs can be chosen
2. After fault injection, ensure that the resulting collision of cipher state can be transferred to the final output, that is, produce the same output.

Collision Fault Attack on ESTATE_TweGIFT-128 and GIFT-COFB.

Figure 5 shows the simplified description of the encryption process of ESTATE_TweGIFT-128 and GIFT-COFB. From Figure 5, we can see that the nonce is encrypted by using TweGIFT-128 or GIFT-128, denoted by E_K , to produce an initial value as the input of the subsequent authentication and encryption process as well as (K, A, M) . By fixing K, A and M , we can satisfy the above two conditions of Collision Fault Attack easily in the nonce-chosen setting. The dotted box in Figure 5 indicates the targetted location of fault attack. It is direct to use the approach of Section 4.1 to recover the key of ESTATE_TweGIFT-128 and GIFT-COFB.

Collision Fault Attack on SUNDAAE-GIFT.

TABLE 7 Possible inputs of Sbox with output difference 0001, 0010, and 0100

Output difference	Input difference	Possible inputs
0001	1001	0111, 1110
	1010	0011, 0101, 1001, 1111
	1011	0001, 1010
	1101	0000, 1101
	1110	0010, 0110, 1000, 1100
	1111	0100, 1011
	0101	1000, 1101
0010	0110	0000, 0010, 0100, 0110
	0111	1011, 1100
	1100	0011, 0101, 1001, 1111
	1101	0111, 1010
	1111	0001, 1110
	1100	0000, 0100, 1000, 1100
	1101	0011, 0110, 1011, 1110
0100	1110	0001, 0111, 1001, 1111
	1111	0010, 0101, 1010, 1101

SUNDAE-GIFT is an authenticated encryption mode with GIFT-128 as the underlying block cipher, denoted as E_K . It includes four versions: SUNDAE-GIFT-96, SUNDAE-GIFT-0, SUNDAE-GIFT-128, and SUNDAE-GIFT-64, with length of nonce 96, 0, 128, and 64, respectively. Figure 6 shows the encryption process of SUNDAE-GIFT, where the initial block $B = b_{127}b_{126}b_{125}b_{124}||0^{124}$,

$$b_{127} = \begin{cases} 0 & \text{if } |A| = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$b_{126} = \begin{cases} 0 & \text{if } |M| = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$b_{125}b_{124} = \begin{cases} 00 & \text{if } |N| = 0 \\ 01 & \text{if } |N| = 64 \\ 10 & \text{if } |N| = 96 \\ 11 & \text{if } |N| = 128 \end{cases}$$

In SUNDAE-GIFT, nonce is regarded as a part of the associated data, that is $A^+ = N||A$. A^+ and plaintext M are divided into many 128-bit blocks, among which $A^+ \rightarrow A_0A_1 \dots A_{l_A-1}$ and A_0 consists of the nonce N and a part of A . The dotted box of Figure 6 indicates the targeted location of fault attack. Because the value of $V = E_K(B)$ is unknown, the condition one of collision fault attack cannot be satisfied. So, our

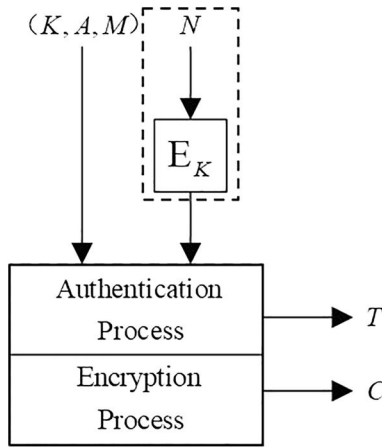


FIGURE 5 Encryption process of ESTATE_TweGIFT-128 and GIFT-COFB

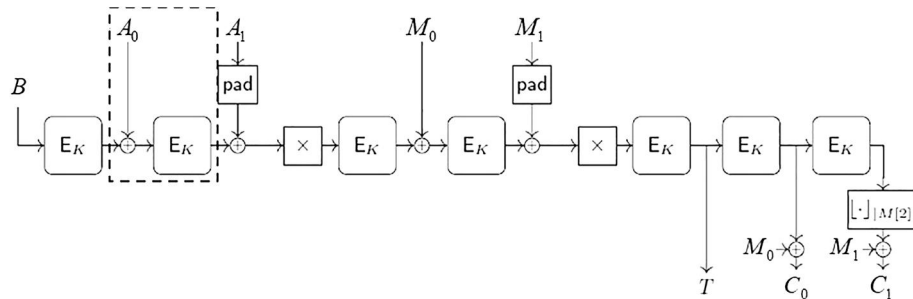


FIGURE 6 Encryption process of SUNDAE-GIFT

first work is to recover the value of V ; assume that B is fixed, which is easy to achieve by choosing appropriate associated data and plaintext. We will show how to recover V by collision fault attack. Here, we regard the tag T as a part of ciphertext C .

4.2.1 | Recover the value of V by collision fault attack

For $0 \leq i \leq 31$, choose 16 pairs of associated data and plaintext (A^{+j}, M^j) ($0 \leq j \leq 15$) satisfying the following: the four bits with serial number $4i, 4i+1, 4i+2, 4i+3$ in A_0 traverses through all values in $\{0,1\}^4$, and the rest part of the associated data and plaintext are identical. In the encryption process of $(A^{+,0}, M^0)$, inject a single-bit fault into the most left bit of the output of $GS^{1,i}$ in $E_K(V \oplus A_0^0)$ (masked with dotted box in Figure 6) to make the bit flip, and get the faulty ciphertext \tilde{C} . Encrypt (A^{+j}, M^j) ($1 \leq j \leq 15$) and get the correct ciphertexts C^1, \dots, C^{15} . Find the integer $g \in \{1, \dots, 15\}$ such that $\tilde{C} = C^g$. For the Sbox $GS^{1,i}$, compute its input difference $\Delta = (A_0^0 \oplus A_0^g)[i]$. According to the possible inputs of $T[\Delta]$ showed in Table 6, get two possible values of $(V \oplus A_0^0)[i]$, which is the input of Sbox $GS^{1,i}$. Then, compute two possible values of $V[i]$. Repeating the foregoing process can uniquely determine the value of $V[i]$. Finally, get the complete $V = V[31]||\dots||V[1]||V[0]$. Through the above approach, we can recover the value of V just using 64 faulty ciphertexts. Then, it is direct to use the way of Section 4.1 to recover the key of SUNDAE-GIFT by injecting fault in the location masked with dotted box.

Table 8 shows the expense to recover key of ESTATE_TweGIFT-128, GIFT-COFB and SUNDAE-GIFT.

We verified the correction of the attack of this section through computation simulations of the collision fault attack on GIFT-128. The simulations were performed thousands times over randomly chosen keys. This a deterministic attack to recover the correct key with fixed number of faulty ciphertexts, which is different with the attack in Section 3.

5 | CONCLUSION

In this study, we research the fault attack on four authenticated encryption modes for GIFT in NIST-LWC. For LOTUS/LO-CUS, we propose a new fault attack technique and recover the

TABLE 8 The expense of collision fault attack on three authenticated encryption modes

Authenticated encryption modes	The number of faulty ciphertexts	Data complexity
ESTATE_TweGIFT-128	64	2^{10}
GIFT-COFB	64	2^{10}
SUNDAE-GIFT	128	2^{11}

correct key with a few faulty ciphertexts. We also point out that the key could be recovered with just one faulty ciphertext if a single-bit permanent fault was injected. Then, we launch the collision fault attack on GIFT algorithm, and subsequently launch the collision fault attack on authenticated encryption modes ESTATE_TweGIFT-128, GIFT-COFT and SUNDAE-GIFT. Our work can recover the key of the authenticated encryption modes effectively, which highlights the need for dedicated fault attack countermeasures in the hardware implementation of the authenticated encryption mode. For nonce-based authenticated encryption modes, we will further search for more effective fault attack techniques to reduce the number of faulty ciphertexts needed to recover the key.

ACKNOWLEDGEMENTS

The authors thank the anonymous reviewers for their helpful comments. This work was supported by the National Natural Science Foundation of China under grant nos. 61572516, 61272041, 61272488.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Shuai Liu  <https://orcid.org/0000-0002-1402-827X>

REFERENCES

- Zussa, L., et al.: Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism. In: Proceedings of the IEEE On-line Testing Symposium-IOLTS 2013Chania, pp. 110–115. Greece (2013)
- Balasch, J., Gierlichs, B., Verbauwhede, I.: An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs. In: Proceedings of the IEEE Fault Diagnosis and Tolerance in Cryptography-FDTC 2011, pp. 105–114. IEEE (2011)
- Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks In: Proceedings of the Cryptographic hardware and Embedded Systems-CHES 2002, pp. 2–12. Redwood Shores (2002)
- Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Proceedings of the Advances in Cryptology-EUROCRYPT'97, pp. 37–51. (1997)
- Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystem. In: Proceedings of the Advances in Cryptology-CRYPTO'97, pp. 513–525. (1997)
- Yen, S., Joye, M.: Checking before output may not be enough against fault-based cryptanalysis. IEEE. Trans. Comput. 49(9), 967–970 (2000)
- Blömer, J., Krummel, V.: Fault based collision attacks on AES. In: Proceedings Fault Diagnosis and Tolerance in Cryptography-FDTC 2006, pp. 106–120. Yokohama (2006)
- Fuhr, T., et al.: Fault attacks on AES with faulty ciphertexts only. In: Proceedings Fault Diagnosis and Tolerance in Cryptography-FDTC 2013, pp. 108–118. (2013)
- Saha, S., et al.: Fault template attacks on block ciphers exploiting fault propagation. In: Proceedings Advances in Cryptology-EUROCRYPT, pp. 162–168. (2020)
- Dobraunig, C., et al.: Statistical ineffective fault attacks on masked AES with fault countermeasures. In: Proceedings Advances in Cryptology-ASIACRYPT 2018, pp. 315–342. (2018)
- Marzouqi, H., Al-Qutayri, M., Salah, K.: Review of gate-level differential power analysis and fault analysis countermeasures. IET Inf Secur. 8(1), 51–66 (2012)
- Simon, T., et al.: Frier: An authenticated encryption scheme with built-in fault detection. In: Proceedings of the Advances in Cryptology-EUROCRYPT 2020, pp. 176–181. (2020)
- Saha, D., Chowdhury, D.R.: Scope: On the side channel vulnerability of releasing unverified plaintexts. In: Proceeding Selected Areas in cryptography-SAC 2015, pp. 417–438. (2015)
- Saha, D., Kuila, S., Chowdhury, D.R.: Escape: Diagonal fault analysis of APE. In: Proceedings of the Progress in Cryptology-INDOCRYPT 2014, pp. 197–216. (2014)
- Dobraunig, C., et al.: Statistical fault attacks on nonce-based authenticated encryption schemes. In: Proceedings of the ASIACRYPT, pp. 369–395. (2016)
- Saha, D., Chowdhury, D.R.: EnCounter: On breaking the nonce barrier in differential fault analysis with a case-study on PAEQ. In: Proceedings of the CHES, pp. 581–601. CHES 2016 (2016)
- Chakraborti, A., et al.: LOTUS-AEAD and LOCUS-AEAD. <https://csrc.nist.gov/projects/lightweight-cryptogr-aphy/round-2-candidates>. Accessed 18 August 2020
- Chakraborti, A., et al.: ESTATE. <https://csrc.nist.gov/projects/lightweight-cryptography/round-2-can-didates>. Accessed 18 August 2020
- Banik, S., et al.: GIFT-COFB v1.0. <https://csrc.nist.gov/projects/lightweight-cryptography/round-2-can-didates>. Accessed 18 August 2020
- Banik, S., et al.: SUNDAE-GIFT v1.0. <https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates>. Accessed 18 August 2020
- Banik, S., et al.: GIFT: A small present. Proceedings Cryptographic hardware and Embedded Systems-CHES 2017, pp. 321–345. (2017)
- Baigneres, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Proceedings of the International Conference on the theory and application of Cryptology and information security, pp. 432–450. (2004)
- Mahri, H.Q.A., et al.: Fault attacks on XEX mode with application to certain authenticated encryption modes. Proceedings of the Australasian Conference on information security and Privacy, pp. 285–305. (2017)
- John, B., Phillip, R.: A block-cipher mode of operation for parallelizable message authentication. Proceedings of the Advances in Cryptology-EUROCRYPT, pp. 384–397. (2002)
- Minematsu, K.: AES-OTR v3.1. <https://competitions.cr.yp.to/round3/aesotrv31.pdf>. Accessed 18 August 2020

How to cite this article: Liu, S., Guan, J., Hu, B.: Fault attacks on authenticated encryption modes for GIFT. IET Inf. Secur. 16(1), 51–63 (2022). <https://doi.org/10.1049/ise2.12041>

APPENDICE

Statistical Fault Attack on GIFT-128

In this section, we research the SFA on GIFT-128. The result shows that SFA on GIFT-128 is not as effective as on AES [8, 15]. At the end, we analyse the reason briefly.

The fault model used here is the same as in [8]. The purpose of the fault injection is to change the distribution of one nibble of GIFT-128.

Fault Model: the stuck at model with a random value e (where e is uniformly distributed in $[0,15]$):

$$\tilde{S}[j] = S[j] \& e$$

where $\&$ denotes bitwise and operation, and \sim denotes faulty state. $S[j]$ denotes a nibble.

Description of the attack: It is assumed that the adversary is able to encrypt unknown plaintexts $P_i (i = 1, \dots, n)$, inject the fault into certain nibble and get corresponding faulty ciphertexts \tilde{C}_i . \tilde{S}_i^{39} denotes the state before Sbox operation of the penultimate round in i th encryption. During every encryption, the fault is injected into j th nibble of \tilde{S}_i^{39} , denoted by $\tilde{S}_i^{39}[j]$.

\tilde{S}_i^{39} can be expressed as a function of \tilde{C}_i and the key of the last round and the penultimate round

$$\tilde{S}_i^{39} = GS^{-1}PB^{-1}(GS^{-1}PB^{-1}(\tilde{C}_i \oplus K^{40}) \oplus K^{39}),$$

therefore, the 4-bit targeted nibble can be deduced using eight key bits of the last round and two key bits of the penultimate round as described in Figure A1. To distinguish the correct key from wrong keys, *Square Euclidean Imbalance* (SEI) is used.

$$SEI(\hat{K}) = \sum_{\delta=0}^{15} \left(\frac{\#\{i | \hat{S}_i^{39}[j] = \delta\}}{n} - \frac{1}{16} \right)^2,$$

where $\hat{K} \in \{0, 1\}^{10}$ denotes the guessed key, $\hat{S}_i^{39}[j]$ denotes the targeted nibble deduced under \hat{K} , and n denotes the number of faulty ciphertexts. This distinguisher assigns high value to the correct key, which leads to non-uniform distributions induced by fault. Therefore, we choose the key with maximal SEI value as the candidate key.

Indeed, it is easy to see that the penultimate round function only performs a permutation on the 4-bit nibble, which does

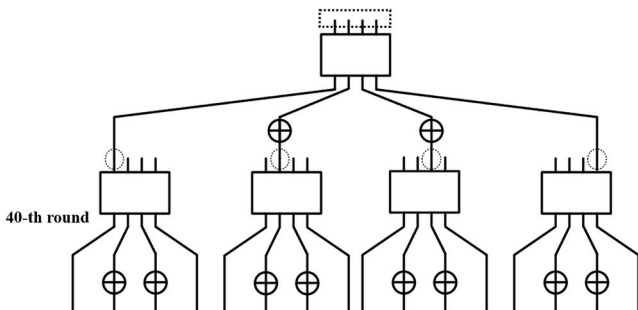


FIGURE A1 Illustration of statistical fault attack on GIFT-128

not modify the distance of the biased distribution from uniform distribution. Hence, we need not consider the value of the two bits of key in the penultimate round. In other words, this allows us to recover 8 bits of the last round key by SFA.

Using the same simulation platform as described in Section 3.5, we run the attack thousand times with different values of n (the number of faulty ciphertexts). Figure A2 displays the result, where the xlabel denotes the number of faulty ciphertexts, and the ylabel denotes the number of keys with SEI value no less than the correct key. Though abundant faulty ciphertexts are used (even when $n = 1000$), there are several wrong keys with SEI value no less than the correct key, that is, the sole key cannot be recovered. We can make a fundamental judgement that there are several 'equivalent' keys which cannot be excluded by SEI.

The main reason why the key cannot be determined uniquely is that the randomness of GIFT-128 round function is not as well as that of AES. So, the nibble derived from several rounds of GIFT-128 is not random enough, which leads to the condition that SEI cannot distinguish the key well.

From Figure A2, the attacker can get about seven candidates by guessing an eight-bit key. It means that we will get about $2^{22.46}$ candidates of the last-round key. Besides the fact that SFA on GIFT-128 needs more faulty ciphertexts and gets too many candidate keys, there is a more crucial problem; we cannot attack the full 128-bit key of GIFT-128 by SFA. According to the Key Schedule of GIFT-128 described in Section 2.1, we need to recover at least two round keys to get the full key. The above approach can be used to attack the 64-bit key of the last round. However, to attack the penultimate-round key by SFA, the attacker needs to get the output of the penultimate round, which is one of the necessary conditions of SFA and is impossible because the last-round key is uncertain.

To sum up, a direct application of SFA on GIFT-128 is not ideal. Therefore, it is significant to launch other fault attacks, such as collision fault attack, to overcome the nonce barrier when GIFT is used as the underlying module of nonce-respecting AE schemes.

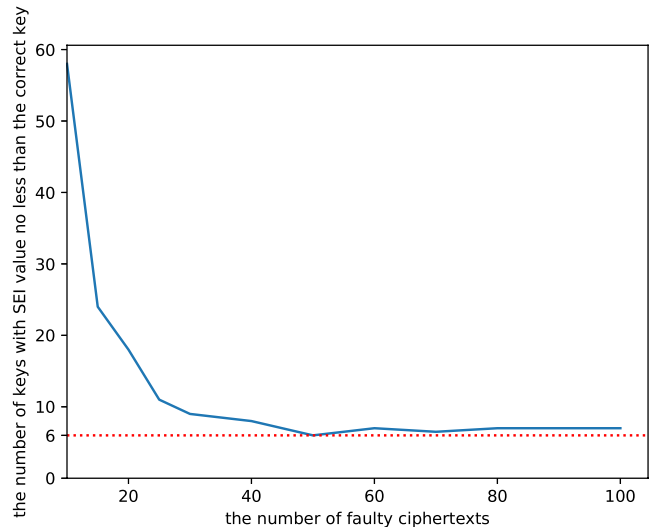


FIGURE A2 The simulation of the statistical fault attack on GIFT-128