Hindawi
Security and Communication Networks
Volume 2019, Article ID 1680263, 12 pages
https://doi.org/10.1155/2019/1680263

Research Article

Random Fault Attacks on a Class of Stream Ciphers

Harry Bartlett,[1] Ed Dawson,[1] Hassan Qahur Al Mahri,[1] Md. Iftekhar Salam,[2] Leonie Simpson,[1] and Kenneth Koon-Ho Wong[1]

[1] *Science and Engineering Faculty, Queensland University of Technology, Brisbane, Australia*
[2] *School of Electrical and Computer Engineering, Xiamen University Malaysia, Selangor, Malaysia*

Correspondence should be addressed to Harry Bartlett; h.bartlett@qut.edu.au

Received 21 December 2018; Accepted 19 May 2019; Published 1 July 2019

Academic Editor: Vincenzo Conti

In this paper, we show that stream ciphers with a particular form of ciphertext output function are vulnerable to differential fault attacks using random faults. The CAESAR competition candidates Tiaoxin-346 and AEGIS-128L both fall into this category, and we show that our attack can be used to recover the secret key of Tiaoxin-346 and the entire state of AEGIS-128L with practical complexity. In the case of AEGIS-128L, the attack can be applied in a ciphertext-only scenario. Our attacks are more practical than previous fault attacks on these ciphers, which assumed bit-flipping faults. Although we also consider other ways of mitigating our attacks, we recommend that cipher designers avoid the form of ciphertext output function that we have identified.

1. Introduction

In this paper, we apply a random fault attack to a particular type of stream cipher. The type of stream cipher we consider is word-based and produces two ciphertext words at each time interval using an output function for each of these ciphertext words that includes both linear and quadratic terms. In particular, one part of the internal state appears as a linear term in the output function producing one of the ciphertext words and in a quadratic term in the output function producing the other ciphertext word. We show that an output function of this type is a straightforward target for random fault attacks.

Two of the third-round candidates in the recent Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), Tiaoxin-346 [1–3] and AEGIS-128L [4, 5], have output functions of the type we describe, with AEGIS-128L being selected in the final portfolio of this competition. Dey et al. [6] noted that the presence of a bitwise AND operation in the output function of these two ciphers makes them vulnerable to bit-flipping fault attacks. We have identified additional features in these two ciphers that permit exploitation using a less restrictive random fault attack. In particular, the following features combine to provide this vulnerability:

(i) Two ciphertext words are generated at each time step

(ii) The function used to compute each of the ciphertext words includes the bitwise AND of two of the underlying internal state words

(iii) The function used to compute one of the ciphertext words also includes the bitwise XOR of one of the state words that was used in the AND of the other ciphertext word.

We treat Tiaoxin-346 and AEGIS-128L as case studies of stream ciphers of this type. In this paper, we demonstrate that a random fault attack is practical against each of these ciphers. This work is an extension of research on Tiaoxin-346 previously published in [7].

1.1. Notation. In this paper, let P, C, and S denote the plaintext, ciphertext, and internal state of the stream cipher, respectively. For encryption, the plaintext P is divided into l_p blocks, with each plaintext block composed of two words. Let P^t denote the plaintext block to be encrypted at time t, with $P^t = P^t[0] \parallel P^t[1]$. Similarly, the ciphertext C^t is given by $C^t = C^t[0] \parallel C^t[1]$, with $C^t[0]$ and $C^t[1]$ denoting the two ciphertext words produced at time t as a result of the encryption of P^t.

Let S^t denote the internal state of the cipher at time t. Suppose the internal state of the cipher consists of n words. Then, $S^t = S^t[0] \parallel S^t[1] \parallel \ldots \parallel S^t[n-1]$. Suppose $S^t[u]$ denotes a particular state word that is common to the output functions of both ciphertext words $C^t[0]$ and $C^t[1]$. The type of output function we consider in this paper has the following form:

$$C^t[0] = l_0(S^t) \oplus q_0(S^t) = l_0^*(S^t) \oplus S^t[u] \oplus q_0(S^t) \quad (1)$$

$$\begin{aligned} C^t[1] &= l_1(S^t) \oplus q_1(S^t) \\ &= l_1(S^t) \oplus (q_1^*(S^t) \otimes S^t[u]), \end{aligned} \quad (2)$$

where $l_0(S^t)$ and $l_1(S^t)$ are linear (XOR) combinations of state words used to form $C^t[0]$ and $C^t[1]$, respectively, and $l_0^*(S^t)$ is the reduced linear combination of state words formed by excluding the state word $S^t[u]$ from $l_0(S^t)$. That is, $l_0(S^t) = l_0^*(S^t) \oplus S^t[u]$. Similarly, $q_0(S^t)$ and $q_1(S^t)$ are (quadratic) functions consisting of the bitwise product (AND) of two state words, and $q_1^*(S^t)$ is the reduced term obtained by excluding the state word $S^t[u]$ from $q_1(S^t)$.

Both of the ciphers we consider use the AES round function [8] in their calculations. In our work, we use Tr to denote one AES round transformation without XOR-ing the subkey; that is,

$$Tr(X) = MixColumns(ShiftRows(SubBytes(X))). \quad (3)$$

Importantly, we note that this transformation is invertible. We will use this property when extending our attack from a single state word to either multiple words or the entire state at a given time point.

2. Fault Attacks

Responses of cryptographic algorithms to faults, either accidental or intentional, differ from one implementation to another. Attacks that exploit the output of an implementation after an error is induced during the device operation are called fault attacks [9]. Boneh et al. [9] first used fault attacks to attack an implementation of RSA. Since then, fault attacks have been widely used against many encryption schemes, including DES [10] and AES [11]. Fault attacks can be very powerful, such that an entire AES key can be retrieved using only a single fault injection [12].

The effect of fault attacks on a cryptosystem depends on

(i) the number of bits affected: one bit, a few bits, one byte, a few bytes, or multiple bytes

(ii) the modification type: stuck-at-zero, stuck-at-one, flip a bit, or a random fault

(iii) fault precision: how precisely the fault location and its timing can be controlled

(iv) fault duration: whether the fault is transient or permanent.

Combinations of these parameters determine the feasibility of a fault attack. For example, if an attack is performed under the assumption that a fault results in flipping every bit in a targeted variable, the assumption implies a very strong adversary. This bit-flipping approach is considered unrealistic in practice [13]. Fault models in which a random fault is injected, such that the adversary does not know in advance the effect of the fault on a variable value, are widely used. Random fault attacks are considered more realistic than bit-flipping fault attacks [13], since less precise control of the fault outcome by the attacker is required.

The fault attacks described in this section can be applied during the encryption phase of any cipher that has the structure described in Section 1. Note that these attacks require an adversary to observe multiple faulty and fault-free ciphertext pairs resulting from the encryption of a given plaintext using the same key and initialization vector. This is a nonce-reuse condition, which may be specifically prohibited by the cipher designer. Such nonce reuse was necessary for the bit-flipping attack by Dey et al. in their attacks on Tiaoxin-346 and AEGIS [6] and similarly is necessary for the random fault attack presented in this paper. The success of these attacks demonstrates the unsuitability of this type of algorithm for use in any environment where nonce reuse may occur.

For simplicity, we first outline the fault attack using bit-flipping faults, taking an approach based on that of Dey et al. in [6]. We then present a fault attack that follows a similar approach but which makes use of the additional ciphertext word produced at each time interval and requires only random faults.

2.1. Bit-Flipping Fault Attacks. Suppose an adversary introduces fault e in state word $S^t[u]$ resulting in the complementation of the entire word. That is, $e = 111\cdots111$ and the faulty state word $S'^t[s]$ is defined as

$$S'^t[s] = S^t[u] \oplus e = \overline{S^t[u]}, \quad (4)$$

where $\overline{S^t[u]}$ denotes the bitwise complement of $S^t[u]$. The fault-free and faulty ciphertext words $C^t[1]$ and $C'^t[1]$ are defined in the following:

$$C^t[1] = l_1(S^t) \oplus (q_1^*(S^t) \otimes S^t[u]) \quad (5)$$

$$C'^t[1] = l_1(S^t) \oplus (q_1^*(S^t) \otimes \overline{S^t[u]}). \quad (6)$$

From (5) and (6), it is clear that $C^t[1]$ XORed with $C'^t[1]$ allows the attacker to recover the value of the term $q_1^*(S^t)$, as shown in the following:

$$\begin{aligned} C^t[1] \oplus C'^t[1] &= (q_1^*(S^t) \otimes S^t[u]) \\ &\quad \oplus (q_1^*(S^t) \otimes \overline{S^t[u]}) \\ &= q_1^*(S^t) \otimes (S^t[u] \oplus \overline{S^t[u]}) = q_1^*(S^t). \end{aligned} \quad (7)$$

2.2. Random Fault Attacks. We adapt the bit-flipping fault attack outlined in Section 2.1 to recover the value of $q_1^*(S^t)$ using only random faults. This is a more realistic assumption than the stringent requirements on the attacker's capabilities needed for bit-flipping fault attacks. Specifically, the inclusion of state word $S^t[u]$ by XOR in the computation of $C^t[0]$ and by AND with $q_1^*(S^t)$ in the computation of $C^t[1]$ (see (1) and (2)) is a vulnerability that can be exploited by an adversary in a random fault attack; specifically, inserting such faults in $S^t[u]$ allows the value of $q_1^*(S^t)$ to be recovered.

In its basic form, this attack provides partial state recovery, but repeated applications may allow full state recovery. In the case studies presented in Sections 3.2 and 4.2 of this paper, we show that partial state recovery (complete recovery of one of three state components) is possible for Tiaoxin-346 and full state recovery is possible for AEGIS-128L.

Suppose now that fault e is a randomly generated value unknown to the attacker. Let the random fault e affect the contents of $S^t[u]$ such that the faulty state word $S'^t[s] = S^t[u] \oplus e$. In this case, the fault-free and faulty ciphertext words $C^t[0]$ and $C'^t[0]$ are determined as shown in the following:

$$C^t[0] = l_0^*\left(S^t\right) \oplus S^t[u] \oplus q_0\left(S^t\right) \tag{8}$$

$$C'^t[0] = l_0^*\left(S^t\right) \oplus S'^t[s] \oplus q_0\left(S^t\right). \tag{9}$$

The attacker can obtain the value of the random error e simply by calculating $C^t[0]$ XOR $C'^t[0]$ (from (8) and (9)), as shown in the following:

$$\begin{aligned} C^t[0] \oplus C'^t[0] &= S^t[u] \oplus S'^t[s] = S^t[u] \oplus S^t[u] \oplus e \\ &= e. \end{aligned} \tag{10}$$

That is, if the attacker has access to the ciphertext pair $(C^t[0], C'^t[0])$, the value of the random error introduced in state word $S^t[u]$ can be obtained from the faulty and fault-free ciphertext alone. Further, we observe that, replacing $\overline{S^t[u]}$ by $S^t[u] \oplus e$, (7) becomes

$$C^t[1] \oplus C'^t[1] = q_1^*\left(S^t\right) \otimes e. \tag{11}$$

Thus, once the random fault value e is known, this can be used to find the value of certain bits in $q_1^*(S^t)$, namely, those bits in $q_1^*(S^t)$ where the corresponding bits of e have a value of one.

This is a limitation of our approach. Unlike the bit-flipping fault attack described by Dey et. al. [6], using random faults does not necessarily recover all of the bits in $q_1^*(S^t)$ with a single fault. However, this limitation can be addressed by performing the random fault injection attack multiple times, injecting a different random fault each time, until all the bits in $q_1^*(S^t)$ are recovered.

The process for recovering the contents of $q_1^*(S^t)$ using random fault injection is summarised in Algorithm 1. In Section 3.2 and Section 4.2, we apply this attack to Tiaoxin-346 and AEGIS-128L, respectively. In the case of Tiaoxin-346, the attack can be repeated six times in total to recover the entire contents of the state component T_6 at time t. Under the known-plaintext scenario (as assumed, for example, by Dey et al), this then allows the secret key to be recovered completely. In the case of AEGIS-128L, the attack can be repeated (under either the known-plaintext or the ciphertext-only scenario) to recover the contents of all state words at time t; from that point, the cipher can then be used to encrypt and verify any variant plaintext that the attacker may wish to send.

3. Case 1: Tiaoxin-346

Tiaoxin-346 is a third-round authenticated encryption stream cipher candidate in the CAESAR cryptographic competition. It uses a 128-bit key K, and a 128-bit initialization vector V. An input plaintext P, of arbitrary length *msglen*, where $0 \le msglen \le 2^{128} - 1$, is encrypted to form ciphertext C, providing confidentiality. The ciphertext C is of the same length as the plaintext P. Integrity assurance is provided by an authentication tag τ of length 128 bits. Tiaoxin-346 also provides integrity assurance for associated data D which does not require confidentiality. Tiaoxin-346 supports associated data of arbitrary length, *adlen*, where $0 \le adlen \le 2^{128} - 1$.

3.1. Structure.

3.1. Structure. Tiaoxin-346 has a similar structure to feedback shift register-based stream ciphers. The internal state has three components T_3, T_4, and T_6 consisting of three, four, and six 128-bit register stages, respectively. This gives the cipher a total internal state size of $13 \times 128 = 1664$ bits.

At the start of the intialisation phase, various stages of the cipher are loaded with copies of the secret key K, the initialisation vector V, and certain specified constants (namely, zero, Z_0, and Z_1). In particular, note that each of the components T_3, T_4, and T_6 has two stages set to the value of K and one stage set to the value of V at this time.

Tiaoxin-346 uses the state update function $Update(T_3^t, T_4^t, T_6^t, M_3^t, M_4^t, M_6^t)$ to update the internal state. Here, M_3, M_4, and M_6 are the external inputs to the states T_3, T_4, and T_6, respectively; depending on the operational phase of the cipher, these may be defined in terms of the defined constants, the associated data, or the plaintext. We note that their values during the initialization and associated data loading phases are determined by constants or associated data, both of which are known to the adversary. More precisley, the state of Tiaoxin-346 at time $t+1$ is defined as

$$T_s^{t+1}[i] = \begin{cases} Tr\left(T_s^t[s-1]\right) \oplus T_s^t[0] \oplus M_s^t & \text{for } i = 0 \\ Tr\left(T_s^t[0]\right) \oplus Z_0 & \text{for } i = 1 \\ T_s^t[i-1] & \text{otherwise,} \end{cases} \tag{12}$$

where $0 \le i \le s-1$ and $s \in \{3, 4, 6\}$. Thus, all stages except for the first two stages in each component are updated by shifting, while the first two stages of each component are updated nonlinearly using the Tr transformation from AES.

3.1.1. Encryption Process. The encryption phase begins after the associated data processing phase is completed. The three components T_3, T_4, and T_6 have been loaded with the associated data at the beginning of the encryption phase. Figure 1 shows the encryption process of Tiaoxin-346. As shown in Figure 1, the plaintext P is divided into two-word blocks $P^t = P^t[0] \parallel P^t[1]$ which are successively loaded into the internal state of each component. Each ciphertext block C^t is then computed after loading the corresponding plaintext block.

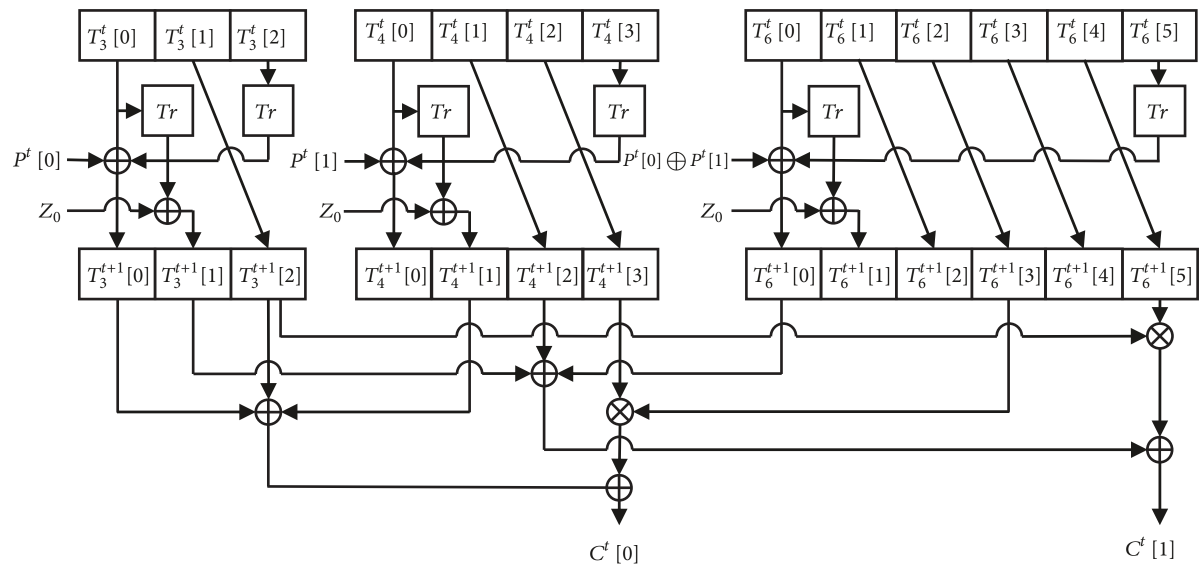


Figure 1: Tiaoxin-346 encryption procedure.

In more detail, the plaintext words are loaded by defining the external inputs M_3, M_4, and M_6 to the state update function as $M_3^t = P^t[0]$, $M_4^t = P^t[1]$, and $M_6^t = P^t[0] \oplus P^t[1]$. The ciphertext words are then computed as

$$C^t[0] = T_3^{t+1}[0] \oplus T_3^{t+1}[2] \oplus T_4^{t+1}[1] \oplus \left(T_6^{t+1}[3] \otimes T_4^{t+1}[3]\right) \quad (13)$$

$$C^t[1] = T_6^{t+1}[0] \oplus T_4^{t+1}[2] \oplus T_3^{t+1}[1] \oplus \left(T_6^{t+1}[5] \otimes T_3^{t+1}[2]\right). \quad (14)$$

Note that C^t depends on P^t, since the plaintext blocks $P^t[0]$ and $P^t[1]$ were XORed into $T_3^{t+1}[0]$ and $T_6^{t+1}[0]$ during the state update process.

3.2. Key Recovery Attack Using Random Faults. The fault attack described in Section 2.2 can be applied during the encryption phase of Tiaoxin-346. Recall that this attack requires an adversary to observe multiple faulty and fault-free ciphertext pairs which are encrypted using the same key and initialization vector.

In our generic attack, the ciphertext words are calculated according to (1) and (2) from Section 1.1. Comparing this general form to (13) and (14), we can make the identification

$$\begin{aligned} S^t[u] &= T_3^{t+1}[2] \\ l_0^*\left(S^t\right) &= T_3^{t+1}[0] \oplus T_4^{t+1}[1] \\ q_0\left(S^t\right) &= T_6^{t+1}[3] \otimes T_4^{t+1}[3] \\ l_1\left(S^t\right) &= T_6^{t+1}[0] \oplus T_4^{t+1}[2] \oplus T_3^{t+1}[1] \\ q_1^*\left(S^t\right) &= T_6^{t+1}[5] \end{aligned} \quad (15)$$

to verify that our attack applies to this cipher. By applying Algorithm 1, we are therefore able to recover the value of $q_1^*(S^t) = T_6^{t+1}[5]$. By repeating this process at consecutive time steps, we can progressively recover the state words $T_6^{t+2}[5], T_6^{t+3}[5], T_6^{t+4}[5], T_6^{t+5}[5]$, and $T_6^{t+6}[5]$. Since $T_6^{t+1}[i] = T_6^t[i-1] =$ for $i = 2, \cdots, 5$ and the invertibility of Tr guarantees that $T_6^{t+1}[0]$ can be uniquely recovered from $T_6^{t+2}[1]$, this enables us to recover the entire contents of T_6^{t+1}. The process for recovering the contents of T_6 with these random faults is shown in Figure 2.

If the plaintext words $P^0[0]$ and $P^0[1]$ are known (as, for example, Dey et al. [6] assume), it is then possible to recover the secret key for Tiaoxin-346. To do this, we first apply the above attack to recover the contents of T_6^1. The state words $T_6^0[0]$ to $T_6^0[4]$ can be recovered from $T_6^1[1]$ to $T_6^1[5]$ by the process described above, while $T_6^0[5]$ can be recovered from $T_6^0[0]$, $T_6^1[0]$, and the message word $M_6^0 = P^0[0] \oplus P^0[1]$. Note again that all the message words used during initialisation are public, so the updates of state component T_6 can be reversed from T_6^0 all the way to the beginning of the initialization phase and the key can then be obtained from the intial loaded contents of $T_6[0]$. Similarly, the initialization vector can be obtained from $T_6[2]$; since this value is public, this provides a confirmation of a successful attack.

From this point, all components in the state of Tiaoxin-346 can be initialized with the known key and IV, and the cipher can then be clocked forwards to encrypt and verify any message chosen by the attacker.

In light of the above comments, the security of this cipher against our key recovery attack depends entirely on maintaining the secrecy of the initial plaintext block. In cases where the message format requires particular header information to be included at the start of the message, this may be problematic.

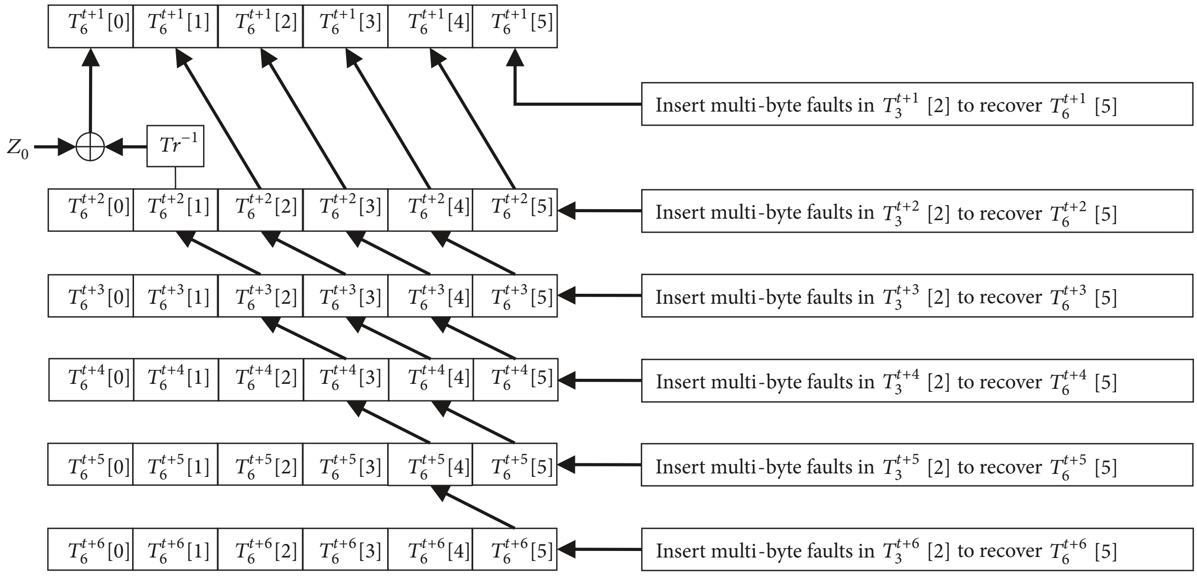

$T_6^{t+1}[0]$ $T_6^{t+1}[1]$ $T_6^{t+1}[2]$ $T_6^{t+1}[3]$ $T_6^{t+1}[4]$ $T_6^{t+1}[5]$
Z_0
Tr^{-1}
Insert multi-byte faults in T_3^{t+1} [2] to recover T_6^{t+1} [5]
$T_6^{t+2}[0]$ $T_6^{t+2}[1]$ $T_6^{t+2}[2]$ $T_6^{t+2}[3]$ $T_6^{t+2}[4]$ $T_6^{t+2}[5]$
Insert multi-byte faults in T_3^{t+2} [2] to recover T_6^{t+2} [5]
$T_6^{t+3}[0]$ $T_6^{t+3}[1]$ $T_6^{t+3}[2]$ $T_6^{t+3}[3]$ $T_6^{t+3}[4]$ $T_6^{t+3}[5]$
Insert multi-byte faults in T_3^{t+3} [2] to recover T_6^{t+3} [5]
$T_6^{t+4}[0]$ $T_6^{t+4}[1]$ $T_6^{t+4}[2]$ $T_6^{t+4}[3]$ $T_6^{t+4}[4]$ $T_6^{t+4}[5]$
Insert multi-byte faults in T_3^{t+4} [2] to recover T_6^{t+4} [5]
$T_6^{t+5}[0]$ $T_6^{t+5}[1]$ $T_6^{t+5}[2]$ $T_6^{t+5}[3]$ $T_6^{t+5}[4]$ $T_6^{t+5}[5]$
Insert multi-byte faults in T_3^{t+5} [2] to recover T_6^{t+5} [5]
$T_6^{t+6}[0]$ $T_6^{t+6}[1]$ $T_6^{t+6}[2]$ $T_6^{t+6}[3]$ $T_6^{t+6}[4]$ $T_6^{t+6}[5]$
Insert multi-byte faults in T_3^{t+6} [2] to recover T_6^{t+6} [5]

Figure 2: Recovering the contents of T_6 by injecting random faults in T_3.

3.3. Experimental Results. Experiments were conducted to analyse the feasibility of the fault attack described in Section 3.2. The experiments were performed as computer simulations using Python 3.6 on a standard desktop computer. The faulty 128-bit words were generated using the Python built-in random number function.

We first investigated the success rate for recovering the state word $T_6^{t+1}[5]$ using multiple random faults. As discussed above, we assume here that the adversary is able to observe multiple faulty and fault-free ciphertext pairs that are encrypted using the same key and initialisation vector. The investigation considered a multibyte fault model; that is, we assume that the error e potentially affects all bytes of the state word $T_3^{t+1}[2]$ in the faulty ciphertext. In the following discussion, "number of faults" refers to the number of faulty ciphertexts of this type observed. Each of the results below is based on 10,000 simulated trials for the given number of faults.

For a given number of random faults, the number of bits recovered by the attack is a random variable. The probability of recovering a specified number of bits increases with the number of faults but decreases as the specified number of bits increases. Table 1 presents the average number of bits recovered for different numbers of faults and the corresponding estimated probability, based on 10,000 trials, of recovering that average number of bits using the given number of faults.

As shown in Table 1, with seven or more faults, an adversary can recover an average of 126 or more of the 128 bits of $T_6^{t+1}[5]$, with a probability higher than 90%. With ten faults, 127 bits can be recovered, with a probability of 99.32%. In all such cases, the adversary needs to guess the remaining bits to recover the entire state word $T_6^{t+1}[5]$.

Table 2 presents the probability of recovering all 128 bits of $T_6^{t+1}[5]$ for a given number of faults. For ten or more faults, the

Table 1: Average Number of Recovered Bits.

Number of Faults	Average Number of Bits Recovered (n)	Probability of Recovering n Bits
1	64	53.71%
2	96	54.61%
3	111	65.86%
4	120	59.57%
5	124	63.45%
6	126	67.63%
7	126	91.88%
8	127	91.07%
9	127	97.49%
10	127	99.32%

Table 2: Success Rate for Recovering All the Bits of $T_6^{t+1}[5]$.

Number of Faults	Success Rate
5	1.73%
6	12.81%
7	37.24%
8	61.72%
9	78.30%
10	87.64%
11	93.69%
12	96.78%
13	98.61%
14	99.16%

```
1: Load key and initialization vector and perform the initialization phase.
2: Encrypt plaintext $P^t$ to compute the fault free ciphertexts $C^t[0]$ and $C^t[1]$.
3: Repeat Step 2 but inject a random multi-byte fault $e$ in the state word $S^t[u]$.
   Proceed to compute the faulty ciphertext words $C'^t[0]$ and $C'^t[1]$.
4: Observe the faulty and fault free ciphertext words and apply Equation (10)
   to recover the value of the random fault $e$.
5: For any bits in the random fault $e$ equal to one, observe the values in the
   corresponding bit positions in the faulty and fault free ciphertext and apply
   Equation (11) to recover the corresponding bits of $q_1^*(S^t)$.
6: Repeat steps 3 to 5 until all of the bits in $q_1^*(S^t)$ are recovered.
```

Algorithm 1: Algorithm for Random Fault Attack on Certain Ciphers.

Table 3: Partial Recovery of $T_6^{t+1}[5]$ and T_6^{t+1} with Different Numbers of Faults.

Recovery of $T_6^{t+1}[5]$			Recovery of T_6^{t+1}			
Number of Faults	Number of Bits to Recover	Success Rate	Number of Faults	Bits to Guess	Guessing Complexity	Success Rate
4	113	99.3%	24	90	2^{90}	95.9%
5	119	99.2%	30	54	2^{54}	95.3%
6	122	99.7%	36	36	2^{36}	98.2%
7	124	99.7%	42	24	2^{24}	98.2%
8	125	99.8%	48	18	2^{18}	98.8%

probability of recovering all bits was over 85%, while 14 faults were required to obtain a 99% probability for recovering all 128 bits of $T_6^{t+1}[5]$.

To explore the trade-off between number of recovered bits and required number of faults, we also investigated the number of bits it was possible to recover with a 99% success rate for specific numbers of random faults. The left-hand side of Table 3 presents the number of bits of $T_6^{t+1}[5]$ that can be recovered for a given number of faults when at least 99% success probability is stipulated, together with the actual success rate observed in each case. For instance, an adversary can recover 124 bits of state $T_6^{t+1}[5]$ with over 99% probability of success by introducing seven faults, and the remaining four bits must then be guessed with a guessing complexity of 2^4.

The right-hand side of Table 3 gives the corresponding results for determining the entire contents of component T_6^{t+1}, including the relevant complexity for guessing the bits that have not already been recovered. Recall that the adversary needs to repeat the same process five more times to recover the state words $T_6^{t+2}[5]$, $T_6^{t+3}[5]$, $T_6^{t+4}[5]$, $T_6^{t+5}[5]$, and $T_6^{t+6}[5]$ in order to achieve this goal. Note from this table that the overall probability of recovering the required number of bits in all six words is over 95% in all cases,and that the complexity of guessing remains practical with 48, 42, or 36 faults. However, the complexity of guessing increases significantly and becomes infeasible for any further reduction in the number of faults beyond 36 faults.

4. Case 2: AEGIS-128L

AEGIS has been selected in the final portfolio of the CAESAR cryptographic competition. AEGIS has three different variants, namely, AEGIS-128, AEGIS-256, and AEGIS-128L. In this section, we discuss AEGIS-128L, as this cipher produces two ciphertext words at each time instant, similar to Tiaoxin-346. This is not the case for AEGIS-128 and AEGIS-256 which only produce one ciphertext word at each time instant.

4.1. Structure. The internal state of AEGIS-128L consists of eight 128-bit register stages $S^t[0], S^t[1] \ldots S^t[7]$ and has a total size of $8 \times 128 = 1024$ bits. The internal state is updated at each time instant using a nonlinear state update function StateUpdate128L(S^t, M_0^t, M_1^t). This update function has two external inputs, M_0^t and M_1^t, and nonlinearity is provided by applying the transformation function Tr to the contents of each register stage, as shown in Figure 3. Under this update function, the state of AEGIS-128L at time $t+1$ is defined as

$$S^{t+1}[i] = \begin{cases} Tr\left(S^t[7]\right) \oplus S^t[0] \oplus M_0^t & \text{for } i = 0 \\ Tr\left(S^t[3]\right) \oplus S^t[4] \oplus M_1^t & \text{for } i = 4 \\ Tr\left(S^t[i-1]\right) \oplus S^t[i] & \text{otherwise,} \end{cases} \quad (16)$$

where $0 \le i \le 7$.

4.1.1. Encryption Process. Figure 3 shows the state update and encryption process of AEGIS-128L. AEGIS-128L performs the encryption phase following the associated data processing phase. As with Tiaoxin-346, each 256-bit plaintext block is split into two words, $P^t = P^t[0] \parallel P^t[1]$, and is encrypted to obtain the corresponding ciphertext block $C^t = C^t[0] \parallel C^t[1]$. This process is repeated at total of l_p times to encrypt the entire plaintext message.

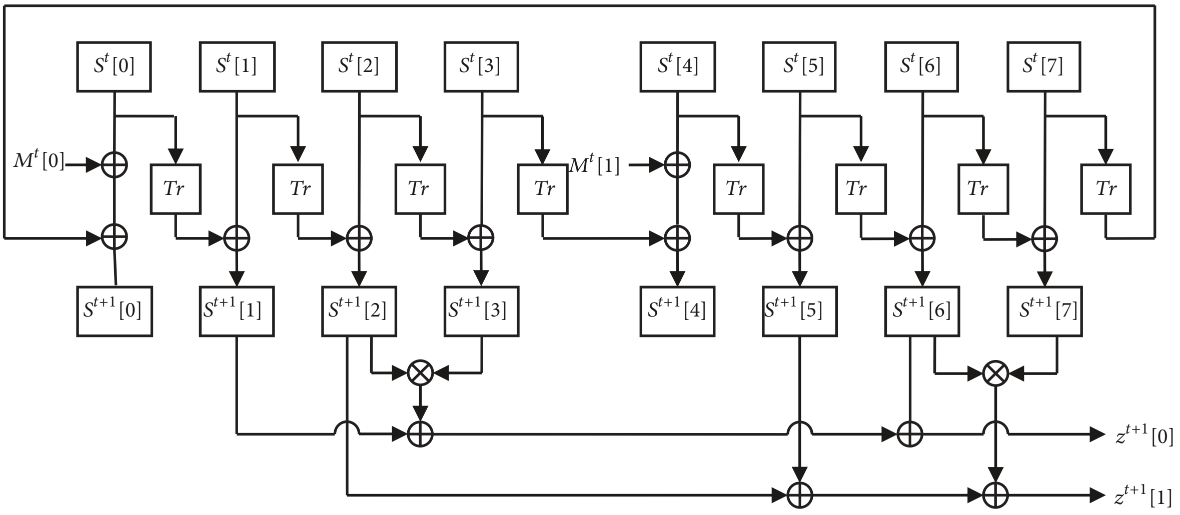


Figure 3: AEGIS-128L state update and encryption.

Encryption begins by computing the keystream words z_0^t and z_1^t as

$$z_0^t = S^t[1] \oplus S^t[6] \oplus \left(S^t[2] \otimes S^t[3]\right) \quad (17)$$

$$z_1^t = S^t[2] \oplus S^t[5] \oplus \left(S^t[6] \otimes S^t[7]\right). \quad (18)$$

Following the keystream computation, the ciphertext words are then computed by XOR-ing the plaintext words with the corresponding keystream words as follows:

$$\begin{aligned} C^t[0] &= P^t[0] \oplus z_0^t \\ &= P^t[0] \oplus S^t[1] \oplus S^t[6] \oplus \left(S^t[2] \otimes S^t[3]\right) \end{aligned} \quad (19)$$

$$\begin{aligned} C^t[1] &= P^t[1] \oplus z_1^t \\ &= P^t[1] \oplus S^t[2] \oplus S^t[5] \oplus \left(S^t[6] \otimes S^t[7]\right). \end{aligned} \quad (20)$$

After the ciphertext has been computed, the internal state of AEGIS-128L is updated using the state update function StateUpdate128L(S^t, $P^t[0]$, $P^t[1]$). That is, the input plaintext words $P^t[0]$ and $P^t[1]$ are used as the external input for the following time step during the encryption phase of AEGIS-128L.

4.2. State Recovery Attack Using Random Faults. We now show that the fault-based attack described in Section 2.2 can also be applied to the encryption phase of AEGIS-128L. Recall again that the ciphertext words in our generic attack are calculated according to (1) and (2) from Section 1.1.

Comparing this general form to (19) and (20), we see that there are two alternative ways of identifying the generic form with these cipher-specific equations. Identifying C_0 and C_1 from the generic form with C_0 and C_1 (respectively) in (19) and (20), we have

$$\begin{aligned} S^t[u] &= S^t[6] \\ l_0^*\left(S^t\right) &= \mathrm{P}^t[0] \oplus S^t[1] \\ q_0\left(S^t\right) &= S^t[2] \otimes S^t[3] \\ l_1\left(S^t\right) &= P^t[1] \oplus S^t[2] \oplus S^t[5] \\ q_1^*\left(S^t\right) &= S^t[7] \end{aligned} \quad (21)$$

which verifies that our attack applies to this cipher. Using this identification in Algorithm 1 enables us to recover the value of $q_1^*(S^t) = S^t[7]$. Alternatively, we can identify C_0 from the generic form with C_1 in (20) and C_1 from the generic form with C_0 in (19), giving the identification

$$\begin{aligned} S^t[u] &= S^t[2] \\ l_0^*\left(S^t\right) &= P^t[1] \oplus S^t[5] \\ q_0\left(S^t\right) &= S^t[6] \otimes S^t[7] \\ l_1\left(S^t\right) &= P^t[0] \oplus S^t[1] \oplus S^t[6] \\ q_1^*\left(S^t\right) &= S^t[3]. \end{aligned} \quad (22)$$

Applying Algorithm 1 with this identification enables us to recover the value of $q_1^*(S^t) = S^t[3]$.

By repeating this process at time $t+1$, we can also obtain the values of $S^{t+1}[7]$ and $S^{t+1}[3]$. Together with the values of $S^t[7]$ and $S^t[3]$, these enable us to determine the values of $S^t[6]$ and $S^t[2]$ from (16) (this is possible because the transformation function Tr in (16) is invertible). If the plaintext words $P^t[0]$, $P^t[1]$, $P^{t+1}[0]$, and $P^{t+1}[1]$ are also known, then the process used by Dey et al. [6] can befollowed

Table 4: Average Number of Recovered Bits.

Number of Faults	Average Number of Bits Recovered (n)	Probability of Recovering n Bits
1	64	53.33%
2	96	55.11%
3	112	56.71%
4	120	59.39%
5	124	62.60%
6	126	67.73%
7	127	73.96%
8	127	91.03%
9	127	97.08%
10	127	99.11%

to obtain the whole state contents at time t (except that the values of $S^{t+1}[7]$ and $S^{t+1}[3]$ no longer need to be calculated, as they are already known).

We note, however, that our attack can also be extended to recover the entire state even when the attacker does not have access to the plaintext: by recovering the values of $S^{t+2}[7]$ and $S^{t+3}[7]$, one can successively determine the values of $S^{t+1}[6]$, $S^{t}[5]$, $S^{t+2}[6]$, $S^{t+1}[5]$, and $S^{t}[4]$; likewise, by recovering $S^{t+2}[3]$ and $S^{t+3}[3]$, we can determine $S^{t}[1]$ and $S^{t}[0]$, so that the entire state contents at time t are known without knowing any of the plaintext. Thus, this attack is stronger than that reported by Dey et al., as it is a ciphertext-only attack.

4.3. Experimental Results. As for Tiaoxin-346, experiments were conducted to analyse the feasibility of the fault attack on AEGIS-128L described in Section 4.2. The experiments were again performed as computer simulations using Python 3.6 on a standard desktop computer and the faulty 128-bit words were generated using the Python built-in random number function.

We began by investigating the success rate for recovering the state word $S^{t}[3]$ using multiple random faults applied to the state word $S^{t}[2]$. As discussed previously, we assume that the adversary is able to observe multiple faulty and fault-free ciphertext pairs that are encrypted using the same key and initialisation vector. The investigation again considered a multibyte fault model; that is, we assume that the error e potentially affects all bytes of the state word $S^{t}[2]$ in the faulty ciphertext. As before, "number of faults" refers to the number of faulty ciphertexts of this type observed and each of the results below is based on 10,000 simulated trials for the given number of faults.

As for Tiaxoin-346, the probability of recovering a specified number of bits increases with the number of faults but decreases as the specified number of bits increases. Table 4 presents the average number of bits recovered for different numbers of faults and the corresponding probability of recovering that average number of bits using the given number of faults.

Table 5: Success Rate for Recovering All the Bits of $S^{t}[3]$ or $S^{t}[7]$.

Number of Faults	Success Rate
5	1.53%
6	13.18%
7	37.01%
8	59.58%
9	78.14%
10	88.35%
11	94.04%
12	96.70%
13	98.38%
14	99.07%

As shown in Table 4, with eight or more faults, an adversary can recover an average of 127 or more of the 128 bits of $S^{t}[3]$, with a probability higher than 90%. With ten faults, an average of 127 bits can be recovered, with a probability of 99.11%. In all such cases, the adversary needs to guess the remaining bit to recover the entire state word $S^{t}[3]$. As the process for recovering bits from $S^{t}[7]$ is identical to that for recovering bits from $S^{t}[3]$, similar probabilities can be expected to apply to the recovery of $S^{t}[7]$.

Table 5 presents the probability of recovering all 128 bits of $S^{t}[3]$ (or all 128 bits of $S^{t}[7]$) for a given number of faults. For ten or more faults, the probability of recovering all bits was over 85%, while 14 faults were required to obtain a 99% probability for recovering all 128 bits of $S^{t}[3]$ or $S^{t}[7]$.

Finally, we again explored the trade-off between number of recovered bits and required number of faults by investigating the number of bits it was possible to recover with a 99% success rate for specific numbers of random faults. The left-hand side of Table 6 presents the number of bits of $S^{t}[3]$ or $S^{t}[7]$ that can be recovered for a given number of faults when at least 99% success probability is stipulated, together with the actual success rate observed in each case. For instance, an adversary can recover 124 bits of the state word $S^{t}[3]$ with over 99% probability of success by introducing seven faults, and the remaining four bits must then be guessed with a guessing complexity of 2^{4}.

The right-hand side of Table 6 gives the corresponding results for determining the entire state contents S^{t}, for each of the attacks discussed in Section 4.2. For the known-plaintext scenario (also contemplated by Dey et al), the adversary needs to repeat the attack process four times in all in order to recover the entire state. Under these conditions, the overall probability of recovering the required number of bits in all four words is over 97% in all cases and the complexity of guessing remains practical for as few as 20 faults. However, the complexity of guessing increases significantly and becomes infeasible for any further reduction in the number of faults beyond 20 faults. For the ciphertext-only attack, the adversary needs to repeat the attack process eight times in all, increasing both the required number of

Table 6: Partial Recovery of $S^t[3]$, $S^t[7]$ and S^t with Different Numbers of Faults.

Recovery of $S^t[3]$ or $S^t[7]$			Recovery of S^t					
			Known Plaintext attack			Ciphertext Only attack		
No. of Faults	No. of Bits to Recover	Success Rate	No. of Faults	Guessing Complexity	Success Rate	No. of Faults	Guessing Complexity	Success Rate
4	113	99.4%	16	2^{60}	97.4%	32	2^{120}	94.9%
5	119	99.4%	20	2^{36}	97.7%	40	2^{72}	95.5%
6	122	99.6%	24	2^{24}	98.6%	48	2^{48}	97.2%
7	124	99.7%	28	2^{16}	98.7%	56	2^{32}	97.5%
8	125	99.8%	32	2^{12}	99.3%	64	2^{24}	98.6%

faults and the associated guessing complexity. In this case, the guessing complexity only remains practical for 64 or 56 faults; however, the overall success rate for both of these choices is again above 97%.

5. Impact of Random Fault Attacks on Tiaoxin-346 and AEGIS-128L

In previous sections, we have considered ciphers with a particular form of encryption function and have shown that these ciphers are vulnerable to fault attacks using random faults. We have shown that this attack enables one or more state words to be recovered in any such cipher. Having noted that the ciphers Tiaoxin-346 and AEGIS-128L each exhibit this structure, we have simulated attacks on both of these ciphers and shown that the secret key of Tiaoxin-346 and the entire state contents of AEGIS-128L can be recovered with practical complexity using fault attacks of this sort. Note that we did not perform our experiments on the hardware implementation of either cipher but instead used computer simulations. However, other researchers [13, 14] have demonstrated that it is feasible to apply this random fault model in the hardware implementation of an algorithm. Therefore, our fault attacks should be practical in both ciphers.

In the following sections, we summarise and compare the outcomes of these attacks for each of these ciphers, before considering possible countermeasures and further generalisations of our approach.

5.1. Summary of Outcomes. The discussions in Sections 3.2 and 4.2 show that it is straightforward to recover the contents of a single state word in either cipher by inducing several multibyte faults to a second state word. In terms of the general ciphertext equations, (1) and (2), introducing multibyte faults to the common state word $S^t[u]$ allows us to recover the state word $q_1^*(S^t)$ in each case. Tables 2 and 5 summarise the probabilities of fully recovering the relevant state word for various numbers of multibyte faults for each of these ciphers. An alternative approach is to recover most of the state word using the approach above and guess the remaining bits. This provides a tradeoff between number of faults required and guessing complexity, as summarised in the left-hand sections of Tables 3 and 6.

In terms of recovering a single state word, the attacks on the two ciphers are essentially similar. However, to be of practical use, the attacks must be extended to recover multiple words. Here, the different overall structures of the two ciphers result in different possible outcomes. In the case of Tiaoxin-346, extending the attack to recover six designated state words allows the whole of component T_6 to be recovered; knowledge of the first plaintext block then allows complete key recovery. This attack can recover the secret key of Tiaoxin-346 using 36 random faults with a practical complexity of guessing, namely, 2^{36}. The attack complexity can be further reduced by increasing the number of faults: for instance, as shown in Table 3, the attack complexity can be reduced to 2^{24} and 2^{18} for 42 and 48 faults, respectively.

In the case of ASEGIS-128L, there are two possible attack scenarios, both of which allow the entire state to be recovered at a specified time point. The more significant of these is a ciphertext only attack, which requires eight state words to be recovered: as shown in the right-hand section of Table 6, this can be done using 56 faults with a guessing complexity of 2^{32} or 64 faults with a guessing complexity of 2^{24}. On the other hand, provided the attacker knows two successive blocks of plaintext, a known plaintext attack is possible. This attack requires only four state words to be recovered using random faults and therefore requires fewer faults to be applied. In fact, 20 faults are sufficient to apply this attack, with an associated guessing complexity of 2^{36}. Again, as shown in the central section of Table 6, the guessing complexity can be lowered by using more faults: for example, with 32 faults, the guessing complexity is reduced to 2^{12}.

In summary, there are two main differences between the full attacks for these two ciphers:

(1) The full attack on Tiaoxin-346 is a known-plaintext attack and requires at least the first block of plaintext to be known to the attacker, while AEGIS-128L can be attacked with ciphertext only.

(2) Given the necessary plaintext, the attack on Tiaoxin-346 allows full key recovery, whereas the attack on AEGIS-128L allows state recovery only.

In terms of the full attacks discussed here, each of the ciphers has specific structural features which allow these attacks to be mounted. We discuss these features and their security implications in Section 5.3 below.

Table 7: Comparison of Our Approach with Existing Approach.

Cipher	Information Available	Reference	Fault Model	No. of Faulty Ciphertexts	Complexity
Tiaoxin-346	Known Plaintext	this work	Random	36	2^{36}
		[6]	Bit Flipping	3	1
AEGIS-128L	Known Plaintext	this work	Random	20	2^{36}
		[6]	Bit Flipping	4	1
	Ciphertext Only	this work	Random	56	2^{32}
		[6]∗	Bit Flipping	8	1

∗ This attack is an obvious extension of the work reported in [6].

5.2. Comparison to Existing Attacks. As discussed previously, Dey et al. [6] also described differential fault attacks on Tiaoxin-346 and AEGIS-128L, but using bit-flipping faults rather than random faults. Table 7 compares the work of Dey et al. [6] with our approach based on random faults.

As shown in Table 7, in each case, our technique requires a much larger number of faults and a comparatively larger complexity when compared to the corresponding attack of Dey et al. [6]. However, our technique works under a random fault model, whereas the other attacks work under a bit-flipping fault model. The assumption of the random fault model is more practical, as the inserted fault e can be any 128-bit value, whereas in the bit-flipping fault model, the fault e must be $e = 111 \cdots 1$. This is a serious restriction of the latter model. The random fault model has been shown to be feasible in actual hardware, whereas the bit-flipping fault model is largely theoretical [13].

Note again that both the fault attacks proposed in this paper and those proposed by Dey et. al. [6] are differential fault attacks; therefore, these attacks require the observation of multiple ciphertexts computed over the same key and initialization vector. This falls under the nonce-reuse scenario, which is prohibited by the designers of these ciphers. The practical nature of our attacks confirms the importance of adhering to this restriction. To the best of our knowledge, there are no other nonce-reuse-based attacks on these ciphers.

5.3. Mitigation Techniques. The ciphers we have considered in this paper produce two ciphertext words at each time step. The key observation that makes our fault attack possible for these ciphers is that the same state word is used in both ciphertext functions in the specific manner described in (1) and (2), namely, appearing linearly in one equation and as part of a quadratic term in the other equation. It is therefore crucial, for any cipher which calculates multiple ciphertext words at each time point, that the state words appearing in a quadratic term within the calculation of one ciphertext word do not also appear linearly in the equation for any other ciphertext word.

Beyond this, each of the ciphers we have considered has specific structural features which allow our attack to be extended from individual state words to full state recovery or key recovery. Addressing these features may also help to improve the security of these ciphers.

In the case of Tiaxion-346, our attack can be extended to a key recovery attack because of two combinations of features. Firstly, the state update function is fully reversible once the external inputs are known and the inputs during the initialization and associated data loading phases are public knowledge: as a consequence, the state update function can be reversed through both the associated data loading and initialization phases. Secondly, each component of the state (T_3, T_4, T_6) is loaded at initialization with a complete copy of the key K and is then updated independently of the other components: in combination with the reversibility of the state update function, this means that an attacker only needs the contents of a single component (in our case, T_6) in order to recover the entire key. By contrast, it is not possible to reverse the state update function of AEGIS-128L without partial knowledge of the earlier state contents. This means that state recovery cannot lead to key recovery for this cipher. AEGIS-128L also takes external input from the secret key K during the 10 updates of the initialization phase, making the process of reversing the state update function even more difficult during this phase.

In Tiaoxin-346, reversibility of the state update function through the initialization phase could be prevented by adopting the strategy used in AEGIS-128L of including the secret key K as an external input during this phase. This would prevent an attacker from converting recovery of an individual state component to full key recovery. Another approach which would prevent full key recovery in this cipher would be to perform the initial key loading in such a way that the key is distributed across all the components and cannot be recovered from the initial state of any single component. Redesigning the state update function so as to mix information between components during update operations might also ensure that an attacker must recover the entire state contents in order to clock backwards and extract the key. However, these latter approaches would not protect against an attacker who was able to recover the entire state; in order to guard against such an attacker, reversibility of the update function must be prevented during the initialization phase.

In the case of AEGIS-128L, key recovery is not possible, but full recovery of the state allows the attacker to run the cipher forward to encrypt and verify any variant plaintext he or she chooses. Recovery of the state during encryption is aided by the location of the state words that can be recovered using random faults: if, for instance, the random fault attack recovered $S^t[4]$ and $S^t[0]$ instead of $S^t[7]$ and $S^t[3]$, it would not be possible to determine the contents of any other state words without knowing the input message words $M_0^t = P^{t-1}[0]$ and $M_1^t = P^{t-1}[1]$. The prevalence of the more easily recovered words $S^t[2]$, $S^t[3]$, $S^t[6]$, and $S^t[7]$ in the keystream equations ((17) and (18)) (note that only one input word in each equation does not belong to this group) also makes it easier to recover other state words in the known plaintext scenario. By contrast, the ciphertext equations for Tiaoxin-346 ((13) and (14)) contain multiple words from different components as well as the words that can be recovered using random faults; because of this structure, it is not possible to recover any contents of components T_3 and T_4 from those of T_6, even when the plaintext is known.

Based on the comments above, it is clear that a ciphertext only state recovery attack would not be possible for AEGIS-128L if the random fault attack were only able to recover the state words $S^t[0]$ and $S^t[4]$. This could be achieved by simply replacing $S^t[3]$ and $S^t[7]$ by $S^t[0]$ and $S^t[4]$ in (17) and (18). This change would also require a known plaintext attacker to recover at least six state words before the final two words could be determined from the keystream equations, increasing the difficulty of the known plaintext attack. It would be better, though, to avoid such attacks entirely by applying the advice in the initial paragraph of this section.

As a further general comment, the success of our full attacks in recovering either the full state or the complete contents of a state component depends on being able to retrieve the contents of state words from previous clocks once the content of the targeted state words are known. For example, in Tiaoxin-346, the information gained from $T_6^{t+1}[5]$ immediately applies to $T_6^t[4]$. Also, in AEGIS-128L, the information gained from attacking $S^t[7]$ and $S^{t+1}[7]$ can be used to compute $S^t[6]$. If the states from previous clocks cannot be easily derived from the fault attack target state $q_1^*(S^t)$, relatively straightfoward state recovery strategies like these would not be possible. An obvious example has already been cited in relation to AEGIS-128L: if only $S^t[0]$ and $S^t[4]$ could be recovered from the attack, it would not be possible to compute the contents of other state words through attacks at successive time points without knowledge of the input words M_0^t and M_1^t. Note also that the state recovery process is deterministic in both ciphers due to Tr being invertible. Having a noninvertible nonlinear component in the ciphertext equations may increase attack complexity, at the expense of other possible weaknesses such as state convergence.

5.4. Generalisations and Future Work. In the cases we have considered above (Tiaoxin-346 and AEGIS-128L), the term $q_1(S^t)$ is a simple quadratic term and the term $q_1^*(S^t)$ is a single state word. Under this scenario, the random fault attack recovers $q_1^*(S^t)$ directly. It might also be possible to extend the form of (1) and (2) to cases where $q_1(S^t)$ is a general nonlinear boolean polynomial. In such cases, applying the random fault attack may still provide partial information about the state contents at time t. For example, if $q_1(S^t)$ is a monomial in more than two state words, we would be able to detect when all words in the term $q_1^*(S^t)$ are 1. For a more general $q_1(S^t)$, bit correlations between state words in $q_1^*(S^t)$ and the output ciphertext word could be exploited to recover the contents of the individual state words.

6. Conclusion

In this paper, we have considered ciphers with a particular form of encryption function, namely, those which produce two ciphertext words at each time step and in which a common state word appears linearly in the calculation of one ciphertext word and as part of a quadratic term in calculating the other ciphertext word. We have shown that these ciphers are vulnerable to a differential fault attack using random faults and that this attack enables one or more state words to be recovered at a given time point. Depending on the particular cipher involved, it may also be possible to extend this attack to recover either the secret key or the entire state contents at a specified time point.

The ciphers Tiaoxin-346 and AEGIS-128L were both third-round candidates in the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), with AEGIS-128L being selected in the final portfolio of this competition. Both of these ciphers have encryption functions of the type discussed above. We have used these ciphers as case studies, demonstrating that our random fault attack can be extended to recover the secret key of Tiaoxin-346 and the entire state contents of AEGIS-128L, with practical complexity in each case. In the case of Tiaoxin-346, the secret key can be recovered with 36 faults and a guessing complexity of 2^{36}, while the entire state of AEGIS-128L can be recovered with 56 faults and a complexity of 2^{32} using ciphertext alone, and with 20 faults and a complexity of 2^{36} in the known plaintext case.

In order to detemine these complexities, we simulated the attacks on these ciphers in software. Although we did not perform our experiments on the hardware implementation of either cipher, other researchers [13, 14] have demonstrated that it is feasible to apply this random fault model in the hardware implementation of an algorithm. Therefore, our fault attacks should be practical in both ciphers. We have also compared our fault attacks on these ciphers against those of Dey et al [6]; although our attacks require more multibyte faults in each case, the random fault model assumed in our attacks is much more practical than the bit-flipping faults assumed by Dey et al in their attacks.

We have also considered the factors which facilitate the extension of our random fault attack to enable full key recovery or state recovery in Tiaoin-346 and AEGIS-128L and have suggested possible countermeasures to prevent these full attacks or make them more difficult. Ultimately, however, the best protection against the random fault attacks we have described is to avoid using an encryption function with the form we have identified above.

Finally, we note again that our attack and that of Dey et al both require the observation of multiple ciphertexts computed over the same key and initialization vector. This falls under the nonce-reuse scenario, which is prohibited by the designers of these ciphers. Once again, the practical nature of our attack confirms the importance of observing this restriction.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] I. Nikolic, "Tiaoxin-346 Version 1.0. CAESAR competition," https://competitions.cr.yp.to/round1/tiaoxinv1.pdf.

[2] I. Nikolic, "Tiaoxin-346 Version 2.0. CAESAR competition," https://competitions.cr.yp.to/round2/tiaoxinv2.pdf.

[3] I. Nikolic, "Tiaoxin-346 Version 2.1. CAESAR competition," https://competitions.cr.yp.to/round3/tiaoxinv21.pdf.

[4] H. Wu and B. Preneel, "AEGIS, A Fast Authenticated Encryption Algorithm (v1) CAESAR competition," https://competitions.cr.yp.to/round1/aegisv1.pdf.

[5] H. Wu and B. Preneel, "AEGIS, A Fast Authenticated Encryption Algorithm (v1.1) CAESAR competition," https://competitions.cr.yp.to/round3/aegisv11.pdf.

[6] P. Dey, R. S. Rohit, S. Sarkar, and A. Adhikari, "Differential Fault Analysis on Tiaoxin and AEGIS Family of Ciphers," in *Security in Computing and Communications - SSCC 201*, P. Mueller, S. Thampi, B. M. Alam, R. Ko, R. Doss, and C. J. Alcaraz, Eds., vol. 625, Springer, Singapore, 2016.

[7] I. Salam, H. Q. Mahri, L. Simpson, H. Bartlett, E. Dawson, and K. K. Wong, "Fault attacks on Tiaoxin-346," in *Proceedings of the the Australasian Computer Science Week (ASCW)*, ACM Digital Library, 2018.

[8] J. Daemen and V. Rijmen, *The Design of Rijndael*, Springer, Berlin, Heidelberg, Germany, 2002.

[9] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology—EUROCRYPT'97*, W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, pp. 37–51, Springer, Berlin, Germany, 1997.

[10] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology — CRYPTO '97*, vol. 1294 of *Lecture Notes in Computer Science*, pp. 513–525, Berlin, Heidelberg, Germany, 1997.

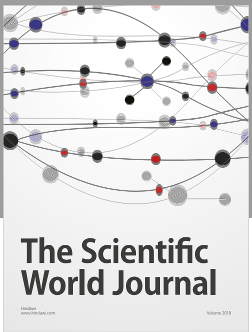
[11] J. Blömer and J. Seifert, "Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)," in *Financial Cryptography*, R. Wright, Ed., vol. 2742 of *Lecture Notes in Computer Science*, pp. 162–181, Springer, Berlin, Heidelberg, Germany, 2003.

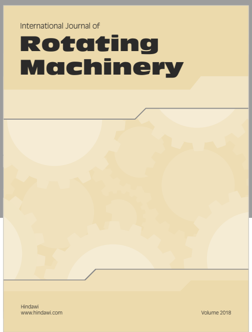
[12] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the Advanced Encryption Standard using a single fault," in *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, C. A. Ardagna and J. Zhou, Eds., vol. 6633 of *Lecture Notes in Computer Science*, pp. 224–233, Springer, Berlin, Heidelberg, Germany, 2011.

[13] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, pp. 3056–3076, 2012.

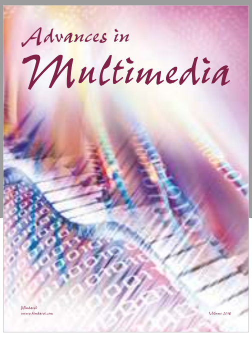
[14] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *Cryptographic Hardware and Embedded System - CHES 2002*, B. S. Kaliski, K. Koç, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, pp. 2–12, Springer, Berlin, Heidelberg, Germany, 2003.

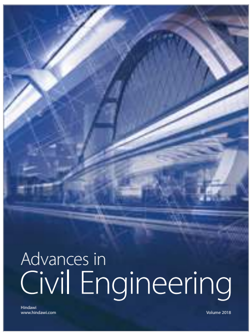
Journal of
Engineering

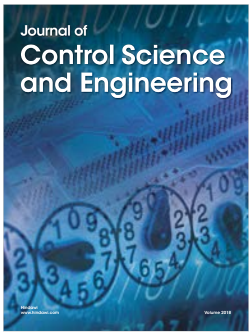

The Scientific
World Journal


International Journal of
Rotating
Machinery

Journal of
Sensors

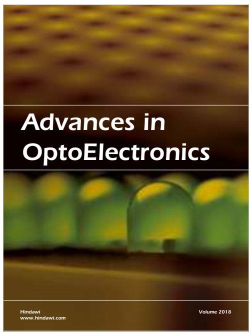

Advances in
Multimedia


Advances in
Civil Engineering


Journal of
Control Science
and Engineering

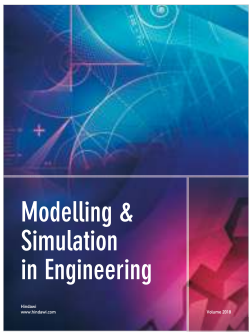
Journal of
Robotics

Journal of
Electrical and Computer
Engineering


Advances in
OptoElectronics

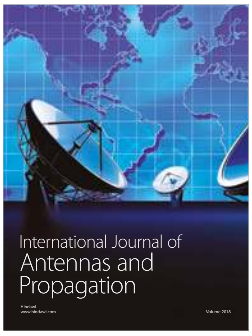
VLSI Design

International Journal of
Navigation and
Observation

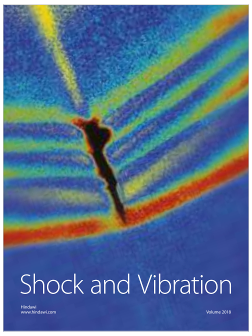

Modelling &
Simulation
in Engineering

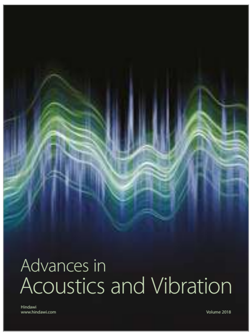
International Journal of
Aerospace
Engineering

International Journal of
Chemical Engineering


International Journal of
Antennas and
Propagation

Active and Passive
Electronic Components


Shock and Vibration


Advances in
Acoustics and Vibration