Differential Fault Analysis on Tiaoxin and AEGIS Family of Ciphers

Prakash Dey[1], Raghvendra Singh Rohit[2], Santanu Sarkar[3], and Avishek Adhikari[1(✉)]

[1] Department of Pure Mathematics, University of Calcutta, Kolkata 700019, India
pdprakashdey@gmail.com, avishek.adh@gmail.com
[2] Department of Mathematics and Statistics, Indian Institute of Science Education and Research, Kolkata, India
iraghvendrarohit@gmail.com
[3] Indian Institute of Technology Madras, Sardar Patel Road, Chennai 600036, India
sarkar.santanu.bir@gmail.com

Abstract. Tiaoxin and AEGIS are two second round candidates of the ongoing CAESAR competition for authenticated encryption. In 2014, Brice Minaud proposed a distinguisher for AEGIS-256 that can be used to recover bits of a partially known message, encrypted 2^{188} times, regardless of the keys used. Also he reported a correlation between AEGIS-128 ciphertexts at rounds i and $i+2$, although the biases would require 2^{140} data to be detected. Apart from that, to the best of our knowledge, there is no known cryptanalysis of AEGIS or Tiaoxin. In this paper we propose differential fault analyses of Tiaoxin and AEGIS family of ciphers in a nonce reuse setting. Analysis shows that the secret key of Tiaoxin can be recovered with 384 single bit faults and the states of AEGIS-128, AEGIS-256 and AEGIS-128L can be recovered respectively with $384, 512$ and 512 single bit faults. Considering multi byte fault, the number of required faults and re-keying reduces 128 times.

Keywords: Stream cipher · AEAD · Differential fault analysis

1 Introduction

Authenticated encryption with associated data (AEAD) is a class of cryptographic primitive for privacy of the plaintext and integrity of both plaintext and associated data. CAESAR [1], a competition for authenticated encryption, is targeting to identify a portfolio of AEAD. Initially, fifty seven authenticated encryptions were submitted to CAESAR. However, in the second round of the competition, 29 submissions survived. Tiaoxin and AEGIS family of ciphers are among the 29 selected second round candidates.

Avishek Adhikari—Research supported in part by National Board for Higher Mathematics, Department of Atomic Energy, Government of India (Grant No. 2/48(10)/2013/NBHM(R.P.)/R&D II/695).

P. Mueller et al. (Eds.): SSCC 2016, CCIS 625, pp. 74–86, 2016.
DOI: 10.1007/978-981-10-2738-3_7

Side channel attacks, such as timing analysis, power analysis and fault analysis, target the implementations of ciphers and test the strength of ciphers in such settings. Power and fault analyses are among the most explored types of side channel attacks.

Biham and Shamir [6] first introduced the idea of Differential Fault Analysis (DFA). Subsequently various symmetric ciphers were analyzed using DFA model. Fault attacks study the robustness of a cryptosystem, in a setting which is in general, weaker than its original or expected mode of operation. In a DFA model, during cipher operations, faults are injected. Since the faults flip the corresponding bits, the attack results in a difference in the state. The resulting faulty output, together with the fault free one, are analyzed to obtain full or a part of the secret information. Although optimistic, this model of attack has been shown to be successful against both stream ciphers and as well as against block ciphers. Most of the proposed ciphers in the eStream portfolio are vulnerable to the fault attacks [3–5,7–14,16,20–22]. AES is also highly vulnerable to fault attacks [2,17,19,23].

Tiaoxin [18] and AEGIS [25] are authenticated cryptographic algorithms submitted to CAESAR by Ivica Nikolić and Hongjun Wu et al. respectively. AEGIS family ciphers were first proposed in SAC 2013 [24]. In SAC 2014, Minaud [15] showed linear biases in AEGIS keystream. However, attack complexity in work of Minaud is higher than the exhaustive key search. There are many similarities in the design principle of Tiaoxin and AEGIS family. Both the ciphers use the same technique of injecting message directly into the state to achieve authentication almost for free. Both ciphers take advantage of AES-NI instructions to achieve outstanding speed in software. The security of both the ciphers relies on the following two assumptions:

A Each Key-IV pair is used to protect only one message.
B If the verification fails, the decrypted plaintext and the wrong authentication tag should not be given as output.

The Tiaoxin and AEGIS designers recommended that Key-IV pair should not be reused. They expressed security concern if all the assumptions are not fulfilled. However, no specific attack was provided. Nevertheless, in the security claims section of the submission document of Tiaoxin, it is stated that

> "If the nonce is reused. Obviously in this case high probability trails (that do not need to end in a zero difference) for the Encryption of Tiaoxin-346 can be used to recover state bytes and to compromise the confidentiality."

Note that one can protect only one message by each Key-IV pair in stream ciphers like Grain. However, there are many papers such as [3,4] on Grain under fault attack where re-key is used. In [1], it is mentioned about fault attack as follows:

> "Sometimes attackers can flip bits in a computation (for example, by firing a laser at a target chip), and deduce secret data from the resulting cipher output."

The aim of this paper is to strengthen the designers claim by describing a fault attack in a nonce reuse setting that allow the complete key recovery for Tiaoxin and complete state recovery for AEGIS family.

Contribution of the paper: The current paper proposes a differential fault attack model on Tiaoxin and the AEGIS family of ciphers when an adversary has precise control on the fault location and fault timing. The attacker injects single bit faults by re-keying each time to obtain particular state blocks. Then after getting a suitable number of state blocks, the entire state is recovered at a known cycle of operation of the cipher. For Tiaoxin, after reversing the state, the secret key can also be recovered. For AEGIS, the recovered complete state could be used for suitable purposes.

Organization of the paper: The rest of the paper is organized in the following way: In Sect. 2 we provide description of Tiaoxin and AEGIS family of ciphers. The attack model considered in this paper and the attacks are described in Sect. 3. Section 3.5 briefly discusses another attack model to reduce the number of faults and re-keying. Finally Sect. 4 concludes the paper.

2 Description of the Ciphers

In this section we briefly describe (only the relevant parts are described) the ciphers Tiaoxin, AEGIS-128, AEGIS-256 and AEGIS-128L. For a descriptive version of the ciphers, the reader may refer to [18,25]. Tiaoxin and the AEGIS family of ciphers extensively use one keyed round of AES. So we describe the one keyed round of AES first.

2.1 AES Round Function

A sequence of 16-bytes will be called a *word*. Let A and B be two words. We denote by $AES(A, B)$, the one keyed round of AES applied to A with B as the subkey (word to AES matrix conversion is the standard one). Thus

$AES(A, B) = \tau(A) \oplus B$ where $\tau(\cdot) = \text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(\cdot)))$.

One should note that the AES operations $\text{MixColumns}(\cdot), \text{ShiftRows}(\cdot)$ and $\text{SubBytes}(\cdot)$ are all invertible. Thus if $\tau(A)$ is known one can obtain A uniquely and efficiently. Also if $AES(A, B)$ i.e. $\tau(A) \oplus B$ and B are both known, one can easily recover A.

2.2 Description of Tiaoxin

Tiaoxin-346 has three states T_3, T_4 and T_6 composed of $3, 4$ and 6 words respectively. The state update mechanism of Tiaoxin uses a round transformation operation $R(T_s, M)$ with state T_s and a word M as input. The output T_s^{new} of $R(T_s, M)$ is the new state and is given by:

$$T_s^{new}[0] = AES(T_s[s-1], T_s[0]) \oplus M,$$
$$T_s^{new}[1] = AES(T_s[0], Z_0),$$

$$T_s^{new}[2] = T_s[1],$$
$$\ldots,$$
$$T_s^{new}[s-1] = T_s[s-2],$$

where Z_0 is a Tiaoxin constant [18].
The state update operation update$(T_3, T_4, T_6, M_0, M_1, M_2)$ takes three additional words M_0, M_1, M_2, i.e.

update : $T_3 \times T_4 \times T_6 \times M_0 \times M_1 \times M_2 \rightarrow T_3 \times T_4 \times T_6$

The function $update(T_3, T_4, T_6, M_0, M_1, M_2)$ is defined as (See Fig. 1):

$$T_3^{new} = R(T_3, M_0); \quad T_3 = T_3^{new}$$
$$T_4^{new} = R(T_4, M_1); \quad T_4 = T_4^{new}$$
$$T_6^{new} = R(T_6, M_2); \quad T_6 = T_6^{new}$$

Tiaoxin ciphertext and tag generation are done in 4 stages: (1) The Initialization (2) Processing the Authenticated Data (3) The Encryption and (4) The Finalization.

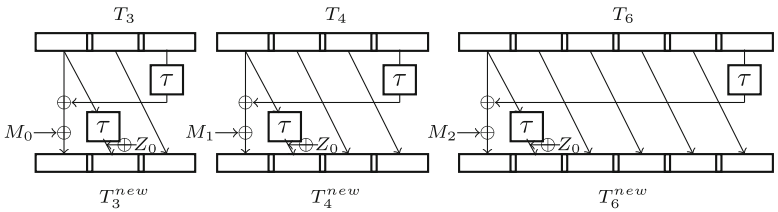

Fig. 1. The *update* function in Tiaoxin-346

After initialization and processing of the authenticated data, in the encryption stage, at each round i, a plaintext $M_i = M_i^0 || M_i^1$, composed of two words M_i^0 and M_i^1, is encrypted to the ciphertext $C_i = C_i^0 || C_i^1$, composed of two words C_i^0 and C_i^1. The encryption at the round i is defined as:

$$\text{update}(T_3, T_4, T_6, M_i^0, M_i^1, M_i^0 \oplus M_i^1)$$
$$C_i^0 = T_3[0] \oplus T_3[2] \oplus T_4[1] \oplus (T_6[3] \& T_4[3]),$$
$$C_i^1 = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus (T_6[5] \& T_3[2])$$

2.3 Description of AEGIS-128

Five 128 bit substates $S_0, \ldots, S_4$ constitutes the inner state of AEGIS-128. Let $S_{i,0}, \ldots, S_{i,4}$ be the substates at the beginning of round i So we have $S_i = S_{i,0} || S_{i,1} || S_{i,2} || S_{i,3} || S_{i,4}$, where each $S_{i,j}$ is a word and $||$ is the concatenation operator.

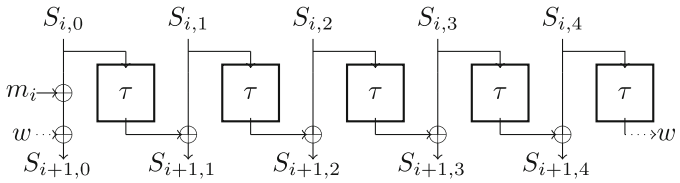

Fig. 2. The state update function of AEGIS-128

At each round i, a 16-byte data block m_i is used to update the state. The new state S_{i+1} is computed as follows:

$$
\begin{aligned}
S_{i+1,0} &= AES(S_{i,4}, S_{i,0} \oplus m_i),\\
S_{i+1,1} &= AES(S_{i,0}, S_{i,1}),\\
S_{i+1,2} &= AES(S_{i,1}, S_{i,2}),\\
S_{i+1,3} &= AES(S_{i,2}, S_{i,3}),\\
S_{i+1,4} &= AES(S_{i,3}, S_{i,4})
\end{aligned}
$$

Figure 2 represents the state update function of AGEIS-128.

AEGIS-128 ciphertext and tag generation are done in 4 stages: (1) The Initialization (2) Processing the Authenticated Data (3) The Encryption and (4) The Finalization.

AEGIS-128 takes a 128 bit key and 128 bit nonce. After initialization and processing of the authenticated data, in the encryption stage, at each round, a 16-byte plaintext block P is used to update the state, and P is encrypted to C as $C = P \oplus z_i$, where $z_i = S_{i,1} \oplus S_{i,4} \oplus (S_{i,2} \& S_{i,3})$ is the 16-byte block keystream.

2.4 Description of AEGIS-256

At the beginning of the i-th round, the (6-word) state of AEGIS-256 is given by $S_i = S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}||S_{i,5}$, where each $S_{i,j}$ is a word. At each round i, a 16-byte data block m_i is used to update the state. The new state S_{i+1} is computed as follows:

$$
\begin{aligned}
S_{i+1,0} &= AES(S_{i,5}, S_{i,0} \oplus m_i),\\
S_{i+1,1} &= AES(S_{i,0}, S_{i,1}),\\
S_{i+1,2} &= AES(S_{i,1}, S_{i,2}),\\
S_{i+1,3} &= AES(S_{i,2}, S_{i,3}),\\
S_{i+1,4} &= AES(S_{i,3}, S_{i,4}),\\
S_{i+1,5} &= AES(S_{i,4}, S_{i,5}).
\end{aligned}
$$

Like AEGIS-128, AEGIS-256 ciphertext and tag generation is also done in 4 stages: (1) The Initialization (2) Processing the Authenticated Data (3) The Encryption and (4) The Finalization.

After initialization and processing of the authenticated data, in the encryption stage, at each round, a 16-byte plaintext block P is used to update the state. Also P is encrypted to C where $C = P \oplus S_{i,1} \oplus S_{i,4} \oplus S_{i,5} \oplus (S_{i,2} \& S_{i,3})$.

2.5 Description of AEGIS-128L

At the beginning of the i-th round, the (8-word) state of AEGIS-128L is given by $S_i = S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}||S_{i,5}||S_{i,6}||S_{i,7}$, where each $S_{i,j}$ is a word. At each round i, two 16-byte data block m_a and m_b are used to update the state. The new state S_{i+1} is computed as follows:

$$
\begin{aligned}
S_{i+1,0} &= AES(S_{i,7}, S_{i,0} \oplus m_a) \\
S_{i+1,1} &= AES(S_{i,0}, S_{i,1}), \\
S_{i+1,2} &= AES(S_{i,1}, S_{i,2}), \\
S_{i+1,3} &= AES(S_{i,2}, S_{i,3}), \\
S_{i+1,4} &= AES(S_{i,3}, S_{i,4} \oplus m_b), \\
S_{i+1,5} &= AES(S_{i,4}, S_{i,5}), \\
S_{i+1,6} &= AES(S_{i,5}, S_{i,6}), \\
S_{i+1,7} &= AES(S_{i,6}, S_{i,7}).
\end{aligned}
$$

AEGIS-128L ciphertext and tag generation are done in 4 stages: (1) The Initialization (2) Processing the Authenticated Data (3) The Encryption and (4) The Finalization.

After initialization and processing of the authenticated data, in the encryption stage, at each round, two 16-byte plaintext block P and P' are used to update the state. Also P and P' are encrypted to C and C' respectively as $C = P \oplus z_{2i}, C' = P' \oplus z_{2i+1}$, where $z_{2i} = S_{i,1} \oplus S_{i,6} \oplus (S_{i,2} \& S_{i,3}), z_{2i+1} = S_{i,2} \oplus S_{i,5} \oplus (S_{i,6} \& S_{i,7})$ are two 16-byte block keystream.

3 Attack Description

The current paper assumes the following attack model:

The attacker can run the cipher with the same secret key, public parameters and plaintext several times. The attacker is able to inject single bit faults. A single bit fault flips the value of the corresponding bit. The attacker has control on the fault timing i.e., the attacker is able to induce single bit fault at any chosen cycle of operation of the cipher. The attacker has control on the fault location i.e., the attacker is able to induce single bit fault at any chosen location. The plaintext and the corresponding normal/faulty ciphertext is available to the attacker.

3.1 Attack on Tiaoxin

Let us consider three consecutive ciphertext generation rounds $i, i+1$ and $i+2$. At round i, the plaintext $M_i = M_i^0 || M_i^1$, composed of two words M_i^0 and M_i^1, is encrypted to the ciphertext $C_i = C_i^0 || C_i^1$ composed of two words C_i^0 and C_i^1 as:

$$\begin{aligned}
&\text{update}(T_3, T_4, T_6, M_i^0, M_i^1, M_i^0 \oplus M_i^1)\\
&C_i^0 = T_3[0] \oplus T_3[2] \oplus T_4[1] \oplus (T_6[3] \& T_4[3]),\\
&C_i^1 = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus (T_6[5] \& T_3[2]).
\end{aligned}$$

We first consider faults at round i. To be precise we inject faults at round i, just after the call to the state update function.

Let us now consider a single bit fault at the r-th bit of the j-th byte of the block $T_6[5]$ i.e., at the r-th bit of the byte $T_6[5][j], 0 \le r \le 7, 0 \le j \le 15$. Due to the fault, the faulty value of $T_6[5][j]$ becomes $T_6[5][j] \oplus f$, where the r-th bit of f is '1', remaining bits being '0's.
Now the fault free ciphertext is given by

$$C_i^1 = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus (T_6[5] \& T_3[2]),$$

whereas its faulty value becomes

$$C_{faulty,i}^1 = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus \big((T_6[5] \oplus F) \& T_3[2]\big),$$

where F is a word with its j-th byte as f, remaining 15 bytes being all 0's. This shows that $C_i^1[j] \oplus C_{faulty,i}^1[j] = f \& T_3[2][j]$. Since $C_i^1[j]$ and $C_{faulty,i}^1[j]$ are both available to the attacker and r-th bit of f is known to being '1', one can recover the r-th bit of the byte $T_3[2][j]$ directly and uniquely.

This shows that, by injecting single bit faults (at each re-keyed run) to the r-th bit of the j-th byte of the block $T_6[5]$ at round i, one can deterministically obtain the r-th bit of the j-th byte of the block $T_3[2]$ for any $0 \le j \le 15$ and $0 \le r \le 7$. Thus with 128 faults to $T_6[5]$, it is possible to recover the entire $T_3[2]$ block. Hence we arrive at the following proposition:

Proposition 1. *Given any ciphertext generation round i, by injecting* 128 *faults to the block $T_6[5]$ one can always recover the block $T_3[2]$.*

Key recovery procedure: We now present the key recovery procedure based on Proposition 1. For that we consider faults at rounds $i, i+1$ and $i+2$.
To avoid ambiguity, we use the superscript i, to denote the state values at round i. For example, with this new notation, T_s^i represents the state T_s at round i.
At round i, by injecting faults to the block $T_6^i[5]$, just after the state update call, one recovers the block $T_3^i[2]$. At round $i+1$, the state T_3 is transformed to

$$T_3^{i+1} = (AES(T_3^i[2], T_3^i[0]) \oplus M_0^i, \quad AES(T_3^i[0], Z_0), \quad T_3^i[1]).$$

Clearly by injecting faults to the blocks $T_6^{i+1}[5]$ and $T_6^{i+2}[5]$ respectively at rounds $i+1$ and $i+2$, just after the state update call, one can recover the

block $T_3^i[1]$ and $AES(T_3^i[0], Z_0)$. Since $AES(T_3^i[0], Z_0)$ and Z_0 are both known, $T_3^i[0]$ can now be recovered. Thus by injecting 3×128 faults at three consecutive ciphertext generation rounds $i, i+1$ and $i+2$ one can recover the entire T_3^i. One should note that,

$$T_3^i = \big(AES(T_3^{i-1}[2], T_3^{i-1}[0]) \oplus M_0^{i-1},\quad AES(T_3^{i-1}[0], Z_0),\quad T_3^{i-1}[1]\big).$$

Thus

$$\begin{aligned} T_3^i[0] &= AES(T_3^{i-1}[2], T_3^{i-1}[0]) \oplus M_0^{i-1},\\ T_3^i[1] &= AES(T_3^{i-1}[0], Z_0),\\ T_3^i[2] &= T_3^{i-1}[1]. \end{aligned}$$

Clearly from T_3^i we can recover T_3^{i-1} i.e., T_3 state update is invertible. Now during the initialization phase, the state T_3 was initialized by (key, key, IV). Thus for Tiaoxin, the secret key can be recovered with 384 single bit faults. The attack strategy for Tiaoxin is illustrated in Fig. 3.

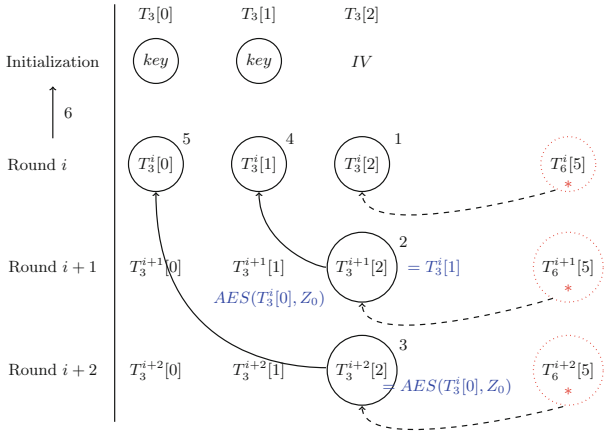

Fig. 3. Attack strategy on Tiaoxin: Here 1,2,... stand for the 1st, 2nd, ... steps of the attack procedure, "$*$" denotes the fault injection, the dotted arrow denotes the consequence of Proposition 1, the arrow from a state T_i to a state T_j denotes that the state T_j can be recovered from the state T_i.

3.2 Attack on AEGIS-128

Let us consider two consecutive ciphertext generation rounds i and $i+1$. Under our attack model both the 16-byte block keystreams z_i and z_{i+1} will be available to the attacker. The state of the cipher at these rounds are given by

$$\begin{aligned} S_i &= S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4},\\ S_{i+1} &= S_{i+1,0}||S_{i+1,1}||S_{i+1,2}||S_{i+1,3}||S_{i+1,4} \end{aligned}$$

and the corresponding 16-byte keystreams are given by

$$z_i = S_{i,1} \oplus S_{i,4} \oplus (S_{i,2} \& S_{i,3}),$$
$$z_{i+1} = S_{i+1,1} \oplus S_{i+1,4} \oplus (S_{i+1,2} \& S_{i+1,3}).$$

As in the case of Tiaoxin, with 128 faults to $S_{i,2}$, it is possible to recover the entire $S_{i,3}$ block. Similarly by injecting 128 faults to $S_{i,3}$, it is possible to recover the entire $S_{i,2}$ block. Thus we arrive at the following proposition:

Proposition 2. *Given any ciphertext generation round i, by injecting* 128 *single bit faults to $S_{i,3}$ (or $S_{i,2}$) one can always recover the block $S_{i,2}$ (or $S_{i,3}$).*

State Recovery Procedure: We now present the state recovery procedure based on Proposition 2. For that we consider faults at rounds i and $i+1$.

By injecting 3×128 single bit faults to $S_{i,3}, S_{i,2}$ and $S_{i+1,3}$ one respectively recovers the blocks $S_{i,2}, S_{i,3}$ and $S_{i+1,2}$. Now $S_{i+1,2} = \tau(S_{i,1}) \oplus S_{i,2}$. Since $S_{i+1,2}$ and $S_{i,2}$ are both known, $S_{i,1}$ can be recovered. Thus from $z_i = S_{i,1} \oplus S_{i,4} \oplus (S_{i,2} \& S_{i,3})$ one can recover $S_{i,4}$. At this moment $S_{i,1}, S_{i,2}, S_{i,3}$ and $S_{i,4}$ are known. Thus one can easily obtain $S_{i+1,3} = \tau(S_{i,2}) \oplus S_{i,3}$ and $S_{i+1,4} = \tau(S_{i,3}) \oplus S_{i,4}$. Now consider $z_{i+1} = S_{i+1,1} \oplus S_{i+1,4} \oplus (S_{i+1,2} \& S_{i+1,3})$ which gives $S_{i+1,1}$ as $S_{i+1,2}$ is also known. Finally $S_{i+1,1} = \tau(S_{i,0}) \oplus S_{i,1}$ gives $S_{i,0}$. Thus with 3×128 faults, we have the state $S_i = S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}$ at the i-th round. The attack strategy on AEGIS-128 is illustrated in Fig. 4.

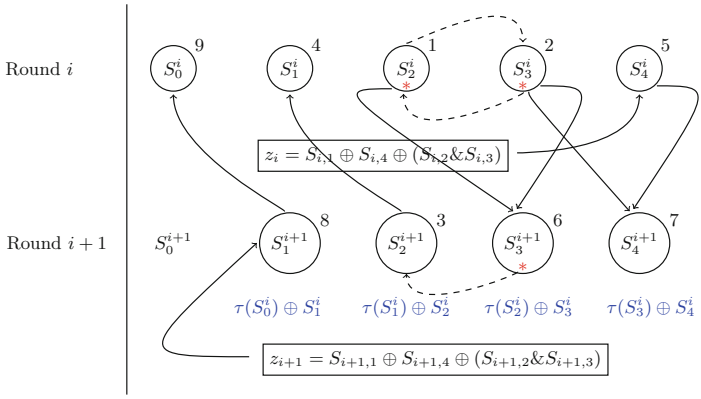

Fig. 4. Attack strategy for AEGIS-128: The notations are similar to that of Fig. 3.

3.3 Attack on AEGIS-256

In this case, we consider three consecutive ciphertext generation rounds $i, i+1$ and $i+2$. Under our attack model the 16-byte block keystreams z_i, z_{i+1} and

z_{i+2} are available to the attacker. The state of the cipher at these rounds are given by

$$\begin{aligned} S_i &= S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}||S_{i,5}, \\ S_{i+1} &= S_{i+1,0}||S_{i+1,1}||S_{i+1,2}||S_{i+1,3}||S_{i+1,4}||S_{i+1,5} \\ S_{i+2} &= S_{i+2,0}||S_{i+2,1}||S_{i+2,2}||S_{i+2,3}||S_{i+2,4}||S_{i+2,5} \end{aligned}$$

and the corresponding 16-byte keystreams are given by

$$\begin{aligned} z_i &= S_{i,1} \oplus S_{i,4} \oplus S_{i,5} \oplus (S_{i,2} \& S_{i,3}), \\ z_{i+1} &= S_{i+1,1} \oplus S_{i+1,4} \oplus S_{i+1,5} \oplus (S_{i+1,2} \& S_{i+1,3}), \\ z_{i+2} &= S_{i+2,1} \oplus S_{i+2,4} \oplus S_{i+2,5} \oplus (S_{i+2,2} \& S_{i+2,3}). \end{aligned}$$

As in AEGIS-128, with 128 faults to $S_{i,2}$, it is possible to recover the entire $S_{i,3}$ block. Similarly by injecting 128 faults to $S_{i,3}$, it is possible to recover the entire $S_{i,2}$ block. Thus we arrive at the following proposition:

Proposition 3. *Given any ciphertext generation round i, by injecting* 128 *faults to $S_{i,3}$ (or $S_{i,2}$) one can always recover the block $S_{i,2}$(or $S_{i,3}$).*

We now present the state recovery procedure based on Proposition 3. For that we consider faults at rounds $i, i+1$ and $i+2$.

By Proposition 3, one obtains $S_{i,2}$ and $S_{i,3}$.$S_{i+1,3}$ is obtained from the relation $S_{i+1,3} = \tau(S_{i,2}) \oplus S_{i,3}$. At round $i+1$, $S_{i+1,2}$ can be recovered by injecting 128 faults to $S_{i+1,3}$. Now $S_{i+1,2} = \tau(S_{i,1}) \oplus S_{i,2}$. Since $S_{i+1,2}$ and $S_{i,2}$ are both known, $S_{i,1}$ can be recovered. At round $i+2$, one follows the same procedure to recover $S_{i+2,2}, S_{i+2,3}, S_{i+1,1}$ and $S_{i,0}$. At this moment four blocks $S_{i,0}, S_{i,1}, S_{i,2}$ and $S_{i,3}$ of i-th round are known. By $z_i = S_{i,1} \oplus S_{i,4} \oplus S_{i,5} \oplus (S_{i,2} \& S_{i,3})$ one knows the value of $S_{i,4} \oplus S_{i,5}$. Now consider

$$\begin{aligned} z_{i+1} =& S_{i+1,1} \oplus S_{i+1,4} \oplus S_{i+1,5} \oplus (S_{i+1,2} \& S_{i+1,3}) \\ =& S_{i+1,1} \oplus \tau(S_{i,3}) \oplus S_{i,4} \oplus \tau(S_{i,4}) \oplus S_{i,5} \oplus (S_{i+1,2} \& S_{i+1,3}). \end{aligned}$$

This gives $S_{i,4}$ as the rest are known. Finally $S_{i,4} \oplus S_{i,5}$ gives $S_{i,5}$. Thus with 4×128 faults, we have the state $S_i = S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}||S_{i,5}$ at the i-th round.

3.4 Attack on AEGIS-128L

We consider two consecutive ciphertext generation rounds i and $i+1$. The state of cipher at these rounds are given by

$$\begin{aligned} S_i &= S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}||S_{i,5}||S_{i,6}||S_{i,7}, \\ S_{i+1} &= S_{i+1,0}||S_{i+1,1}||\,S_{i+1,2}||S_{i+1,3}||S_{i+1,4}||S_{i+1,5}||S_{i+1,6}||S_{i+1,7}. \end{aligned}$$

The corresponding known 16-byte keystreams are given by

$$\begin{aligned} z_{2i} &= S_{i,1} \oplus S_{i,6} \oplus (S_{i,2} \& S_{i,3}), \\ z_{2i+1} &= S_{i,2} \oplus S_{i,5} \oplus (S_{i,6} \& S_{i,7}), \\ z_{2i+2} &= S_{i+1,1} \oplus S_{i+1,6} \oplus (S_{i+1,2} \& S_{i+1,3}), \\ z_{2i+3} &= S_{i+1,2} \oplus S_{i+1,5} \oplus (S_{i+1,6} \& S_{i+1,7}). \end{aligned}$$

For AEGIS-128L we have the next proposition similar to Proposition 3.

Proposition 4. *Given any ciphertext generation round i, by injecting* 128 *faults to each of $S_{i,3}, S_{i,2}, S_{i,7}$ and $S_{i,6}$ one can always recover the blocks $S_{i,2}, S_{i,3}, S_{i,6}$ and $S_{i,7}$ respectively.*

We now present the state recovery procedure based on Proposition 4.

By injecting 4×128 single bit faults, one obtains $S_{i,2}, S_{i,3}, S_{i,6}$ and $S_{i,7}$.z_{2i} and z_{2i+1} respectively give $S_{i,1}$ and $S_{i,5}$. Now $S_{i+1,1}$ and $S_{i+1,5}$ are recovered by considering the following relations

$$\begin{aligned} S_{i+1,2} &= \tau(S_{i,1}) \oplus S_{i,2}, \\ S_{i+1,3} &= \tau(S_{i,2}) \oplus S_{i,3}, \\ S_{i+1,6} &= \tau(S_{i,5}) \oplus S_{i,6}, \\ S_{i+1,7} &= \tau(S_{i,6}) \oplus S_{i,7}, \\ z_{2i+2} &= S_{i+1,1} \oplus S_{i+1,6} \oplus (S_{i+1,2} \& S_{i+1,3}), \\ z_{2i+3} &= S_{i+1,2} \oplus S_{i+1,5} \oplus (S_{i+1,6} \& S_{i+1,7}. \end{aligned}$$

Thus, $S_{i+1,1} = \tau(S_{i,0}) \oplus S_{i,1}$ and $S_{i+1,5} = \tau(S_{i,4}) \oplus S_{i,5}$ respectively give $S_{i,0}$ and $S_{i,4}$. This shows that with 4×128 faults, we have the state $S_i = S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}||S_{i,5}||S_{i,6}||S_{i,7}$ at the i-th round.

3.5 Reducing the Number of Re-Keying

Due to the nature of the ciphers, the attacker can reduce the number of re-keying of the ciphers by injecting faults parallely. For this we consider another

Table 1. Summary of attacks

Cipher	Encryption round	Number of times Key-IV is repeated (Single bit fault model)	Number of times Key-IV is repeated (Multi byte fault model)
Tiaoxin	$i, i+1, i+2$	384	3
AEGIS-128	$i, i+1$	384	3
AEGIS-256	$i, i+1, i+2$	512	4
AEGIS-128L	$i, i+1$	512	4

fault model where the attacker can inject single bit faults to all the 128 bits of a 16-byte block at a time. In this case the number of re-keying will reduce by 128 times. With this fault model the attacker will now respectively require only $3, 3, 4$ and 4 re-keying for Tiaoxin, AEGIS-128, AEGIS-256 and AEGIS-128L. The injected fault being visualized as a multi byte fault. We summarize the attacks in Table 1.

4 Conclusion

In this paper we presented a differential fault analysis on Tiaoxin and the AEGIS family of ciphers. We show one can find the key of Tiaoxin by injecting 384 single bit faults. Also we prove one needs 384 (respectively 512 and 512) single bit faults for AEGIS 128 (respectively AEGIS 256 and AEGIS-128L) to find the state. Reducing the number of single bit faults in an attack model where the adversary does not have the control over the fault injection timing as well as the fault injection location, could be a challenging future work.

References

1. CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) (2013). http://competitions.cr.yp.to/caesar.html
2. Ali, S.S., Mukhopadhyay, D.: A differential fault analysis on AES key schedule using single fault. In: 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 35–42. IEEE (2011)
3. Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the grain family of stream ciphers. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 122–139. Springer, Heidelberg (2012)
4. Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the grain family under reasonable assumptions. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 191–208. Springer, Heidelberg (2012)
5. Berzati, A., Canovas, C., Castagnos, G., Debraize, B., Goubin, L., Gouget, A., Paillier, P., Salgado, S.: Fault analysis of GRAIN-128. In: IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, pp. 7–14. IEEE (2009)
6. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
7. Dey, P., Adhikari, A.: Improved multi-bit differential fault analysis of Trivium. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 37–52. Springer International Publishing, Heidelberg (2014)
8. Dey, P., Chakraborty, A., Adhikari, A., Mukhopadhyay, D.: Improved practical differential fault analysis of grain-128. In: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, pp. 459–464. EDA Consortium, San Jose (2015)
9. Dey, P., Rohit, R.S., Adhikari, A.: Full key recovery of acorn with a single fault. J. Inf. Secur. Appl. **29**, 57–64 (2016)
10. Hojsík, M., Rudolf, B.: Differential fault analysis of Trivium. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 158–172. Springer, Heidelberg (2008)

11. Hojsík, M., Rudolf, B.: Floating fault analysis of Trivium. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 239–250. Springer, Heidelberg (2008)
12. Yupu, H., Gao, J., Liu, Q., Zhang, Y.: Fault analysis of Trivium. Des. Codes Crypt. **62**(3), 289–311 (2012)
13. Karmakar, S., Roy Chowdhury, D.: Fault analysis of grain-128 by targeting NFSR. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 298–315. Springer, Heidelberg (2011)
14. Kircanski, A., Youssef, A.M.: Differential fault analysis of rabbit. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 197–214. Springer, Heidelberg (2009)
15. Minaud, B.: Linear biases in AEGIS keystream. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 290–305. Springer, Heidelberg (2014)
16. Mohamed, M.S.E., Bulygin, S., Buchmann, J.: Using SAT solving to improve differential fault analysis of Trivium. In: Kim, T., Adeli, H., Robles, R.J., Balitanas, M. (eds.) ISA 2011. CCIS, vol. 200, pp. 62–71. Springer, Heidelberg (2011)
17. Mukhopadhyay, D.: An improved fault based attack of the advanced encryption standard. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 421–434. Springer, Heidelberg (2009)
18. Nikolić, I.: Tiaoxin - 346 (2014). http://competitions.cr.yp.to/round1/tiaoxinv1.pdf
19. Saha, D., Mukhopadhyay, D., Chowdhury, D.R.: A diagonal fault attack on the advanced encryption standard. IACR Cryptol. ePrint Arch. **2009**, 581 (2009)
20. Esmaeili Salehani, Y., Kircanski, A., Youssef, A.: Differential fault analysis of Sosemanuk. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 316–331. Springer, Heidelberg (2011)
21. Sarkar, S., Banik, S., Maitra, S.: Differential fault attack against grain family with very few faults and minimal assumptions. IACR Cryptol. ePrint Arch. **2013**, 494 (2013)
22. Sarkar, S., Dey, P., Adhikari, A., Maitra, S.: Probabilistic signature based generalized framework for differential fault analysis of stream ciphers. Cryptogr. Commun. 1–21 (2016)
23. Tunstall, M., Mukhopadhyay, D., Ali, S.: Differential fault analysis of the advanced encryption standard using a single fault. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 224–233. Springer, Heidelberg (2011)
24. Hongjun, W., Bart Preneel, A.: A fast authenticated encryption algorithm. In: 20th International Conference on Selected Areas in Cryptography - SAC 2013, Burnaby, BC, Canada, 14–16 August 2013, Revised Selected Papers, pp. 185–201 (2013)
25. Hongjun, W., Bart Preneel, A.: A fast authenticated encryption algorithm (v1). CAESAR Submission, updated from Cryptology ePrint Archive Report 2013/695, updated from SAC 2013 version (2014). http://competitions.cr.yp.to/round1/aegisv1.pdf