IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 0, No. 0, pp. 0–0. DOI:10.46586/tosc.v0.i0.0-0

Weak Keys in Reduced AEGIS and Tiaoxin

Fukang Liu[1,2], Takanori Isobe[2,3,4], Willi Meier[5], Kosei Sakamoto[2]

[1] East China Normal University, Shanghai, China
liufukangs@163.com
[2] University of Hyogo, Hyogo, Japan
[3] National Institute of Information and Communications Technology, Tokyo, Japan
[4] PRESTO, Japan Science and Technology Agency, Tokyo, Japan
takanori.isobe@ai.u-hyogo.ac.jp, k.sakamoto0728@gmail.com
[5] University of Applied Sciences and Arts Northwestern Switzerland, Windisch, Switzerland
willimeier48@gmail.com

Abstract. AEGIS-128 and Tiaoxin-346 (Tiaoxin for short) are two AES-based primitives submitted to the CAESAR competition. Among them, AEGIS-128 has been selected in the final portfolio for high-performance applications, while Tiaoxin is a third-round candidate. Although both primitives adopt a stream cipher based design, they are quite different from the well-known bit-oriented stream ciphers like Trivium and the Grain family. Their common feature consists in the round update function, where the state is divided into several 128-bit words and each word has the option to pass through an AES round or not. During the 6-year CAESAR competition, it is surprising that for both primitives there is no third-party cryptanalysis of the initialization phase. Due to the similarities in both primitives, we are motivated to investigate whether there is a common way to evaluate the security of their initialization phases. Our technical contribution is to write the expressions of the internal states in terms of the nonce and the key by treating a 128-bit word as a unit and then carefully study how to simplify these expressions by adding proper conditions. As a result, we find that there are several groups of weak keys with 2^{96} keys each in 5-round AEGIS-128 and 8-round Tiaoxin, which allows us to construct integral distinguishers with time complexity 2^{32} and data complexity 2^{32}. Based on the distinguisher, the time complexity to recover the weak key is 2^{72} for 5-round AEGIS-128. However, the weak key recovery attack on 8-round Tiaoxin will require the usage of a weak constant occurring with probability 2^{-32}. All the attacks reach half of the total number of initialization rounds. We expect that this work can advance the understanding of the designs similar to AEGIS and Tiaoxin.

Keywords: AES · AEGIS · Tiaoxin · weak key · distinguisher · key-recovery

1 Introduction

Strong diffusion and confusion are two principles to design secure symmetric-key primitives. It is undoubtable that for almost all symmetric-key primitives the attackers lose the capability to write the accurate boolean expressions of the output bits in terms of the input bits. To address this problem, the cube attack [DS09] was invented to capture partial information of the boolean expressions of the output bits. Especially, with the evolvement of the technique called division property [Tod15, TM16], there exist automatical tools [XZBL16, WHG+19, HLM+20, HSWW20] to search for the desired partial information of the boolean expressions. An evident advantage to utilize the division property with the automatical tools is that attackers can find integral distinguishers [KW02] in a relatively easy way. However, it seems that a naive implementation can only find integral distinguishers holding for all secret keys.

When it comes to the weak-key setting, without knowing what the weak key is in advance, the naive implementation of the division property will obviously fail in finding the integral distinguishers not holding for all keys. How to identify a set of weak keys is nontrivial as there may exist different perspectives of what a weak key should be like and how a weak key influences the attack, which can be seen from the development of the invariant subspace attack [LAAZ11, LMR15, TLS16, GJN+16, Bey18], a popular attack on lightweight symmetric-key primitives in the weak-key setting. In general, the time complexity of a reasonable distinguisher and key-recovery attack in the weak-key setting should be smaller than the number of weak keys.

This work will focus on the integral attack [KW02] on round-reduced AEGIS-128 and Tiaoxin in the weak-key setting. AEGIS-128 [WP13] and Tiaoxin [Nik] are two authenticated encryption (AE) schemes adopting a stream cipher based design. Their common feature consists in the round update function, where the internal state is divided into several 128-bit words and each word has the option to pass through an AES round or not. For AEGIS-128, all the state words will independently pass through the AES round function in one-round update, while only partial state words will independently pass through the AES round function in one-round update for Tiaoxin. Designing a round update function in this way will allow parallel calls to AES-NI, which is a set of instructions for the AES round function designed by Intel. Consequently, the round update functions of both primitives are rather efficient even with several AES rounds. Notably, Jean and Nikolić generalized the way to construct such round update functions in FSE 2016 [JN16], which has attracted the interest of the community.

To improve the unit time to process a 128-bit message block, after the initialization phase and processing the associated data, only one-round update is used to compute each ciphertext block. To ensure the security of the encryption phase, the state sizes of both AEGIS-128 and Tiaoxin are large and the output is computed based on a quadratic boolean function in terms of several state words, which can prevent attackers from recovering the whole secret internal state with the output faster than an exhaustive key search. Moreover, such a way to generate the output also makes reversing the round update function impossible without guessing many state words.

For such a way to compute the output, i.e. the keystream, it has been pointed out by Minaud that there exists a linear bias in the keystream [Min14] soon after the publication of AEGIS. Especially, AEGIS-256 was shown to be insecure against this statistical attack. Later, this idea was applied to MORUS in ASIACRYPT 2018 [AEL+18] and how to construct a model to automatically search for the linear bias in the keystream was also proposed in CRYPTO 2019 [SSS+19]. Such a model was then adapted to re-evaluate the keystream of AEGIS [ENP19]. Although a better linear bias was found for AEGIS-256, both AEGIS-128 and AEGIS-128L remain secure in their keystream generation phase [ENP19].

For AEGIS-128 and Tiaoxin, when the associated data is empty, the phase to process the associated data will be skipped and the attacker can immediately observe the information of the internal state at the encryption phase. As only one-round update is utilized to process each message block in the encryption phase, it is obvious that the security of the initialization phase of both primitives controls the security of the whole AE scheme. Although the designers made an initial study of the initialization phase, only the resistance against the differential attack was investigated. For AEGIS-128, the designers claimed that there are 50 AES rounds in initialization and a difference in the controllable input will pass through more than 10 AES rounds [WP13]. For Tiaoxin, the designers claimed that 6 rounds are sufficient to resist against the differential attack by counting the number of active S-boxes [Nik].

The designers only took the conventional differential attack into account but the feasibility to apply the well-known integral distinguisher on 4-round AES was not discussed.

Surprisingly, there is no third-party cryptanalysis[1] of the initialization phase for both primitives even until now. To fill in the gap, we are motivated to investigate the possibility to utilize the well-known integral distinguisher on 4-round AES [DR02] to analyze their initialization phases.

Our contributions. Due to the fact that the state words are independently processed via the AES round function and the diffusion between the state words is rather weak in the round update function, we find it more suitable and feasible to write the expressions of the internal states in terms of the input by treating a 128-bit word rather than a bit as a unit. The reason to consider the integral distinguisher is also simple as it allows to study the integral property for each term in the expression independently, which fits very well with the way to generate the output for both AEGIS-128 and Tiaoxin. However, other attacks may rely on the interaction between the terms.

To study the integral property for the output, it is essential to study some unusual integral properties that will never appear in real AES but will appear in AEGIS-128 and Tiaoxin. Specifically, we will prove that for some unusual combinations of the AES round function, in the multiset of the outputs for a certain structure of inputs, the same value will appear an even number of times. Without noticing this property, one may add redundant conditions to obtain the integral distinguishers on reduced AEGIS-128 and Tiaoxin or even fail in finding them.

After writing the expressions, analyzing them and adding proper conditions to simplify them are crucial steps to identify the integral distinguishers. It is common in symmetric-key cryptanalysis to add conditions to control the propagation in variables, which has lead to the powerful collision attacks on the MD-SHA hash family [WLF+05, WY05, WYY05], the conditional differential attack [BB93, KMN10] and the conditional cube attack [HWX+17], though the conditions are carefully derived from the bit level. In our attack, the conditions are derived from the byte level and they are hidden in the expressions. It can also be found that the conditions on the key occur for very different reasons in AEGIS-128 and Tiaoxin.

Benefiting from writing the expressions, we observed the feasibility to mount a key-recovery attack by introducing a new variable to represent the output of one AES round for a certain 128-bit word, which is equivalent to appending a round for key recovery. Especially for Tiaoxin, there seems to be one useless round.

As a result, we identified several sets of weak keys with 2^{96} keys each for 5-round AEGIS-128 and 8-round Tiaoxin. For each set of weak keys, the time complexity and data complexity to construct the integral distinguisher are both 2^{32}. The weak key recovery for 5-round AEGIS-128 requires 2^{32} data, 2^{25} memory and 2^{72} time. For 8-round Tiaoxin, the weak key recovery will require the usage of a weak constant occurring with probability 2^{-32}. In addition, the size of each set of weak keys in 8-round Tiaoxin will be reduced to 2^{72} in the key-recovery attack. If a weak constant is used, the key-recovery attack on 8-round Tiaoxin will require 2^{32} data, 2^{24} memory and 2^{48} time. The results are summarized in Table 1.

Organization. The paper is organized in the following way. In Section 2, we will introduce some necessary notations and the specification of the initialization phases of AEGIS-128 and Tiaoxin. Then some new integral properties of the AES round function will be discussed in Section 3. In Section 4 and Section 5, how to derive the distinguishers and key-recovery attacks for 5-round AEGIS-128 and 8-round Tiaoxin will be detailed, respectively. Then, we summarize the attacks on AEGIS-128 and Tiaoxin and discuss the usage of division property in Section 6. Finally, the paper in concluded in Section 7.

[1]There are two existing works [ZXL17, VV18] for Tiaoxin, but neither of them is relative to our attack.

Table 1: The results of the analysis of reduced AEGIS-128 and Tiaoxin in the weak-key setting, where M/T/D represent the memory/time/data complexity, respectively. In addition, the column named "Size" represents the size of each set of weak keys. The column named "#Classes" represents the number of sets of weak keys. The column named "Constant" represents the requirement on the constant. "-" represents negligible.

Target	Attack Type	Rounds	M	T	D	Size	#Classes	Constant
AEGIS-128	Distinguisher	5/10	-	2^{32}	2^{32}	2^{96}	4	arbitrary
Tiaoxin	Distinguisher	8/16	-	2^{32}	2^{32}	2^{96}	12	arbitrary
AEGIS-128	Key recovery	5/10	2^{25}	2^{72}	2^{32}	2^{96}	4	arbitrary
Tiaoxin	Key recovery	8/16	2^{24}	2^{48}	2^{32}	2^{72}	4	weak

2 Preliminaries

In this section, we explain the notations in this paper and briefly describe the initialization phase of AEGIS-128 and Tiaoxin, respectively.

2.1 Notation

1. SB, SR, MC and AC represent the SubBytes, ShiftRows, MixColumns and Constant Addition operations defined in AES, respectively.
2. $\mathsf{MC}^{-1}(X)$ represents the application of the inverse of the Mixcolumns operation of AES to a 128-bit value X.
3. $A(X)$ represents $\mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}(X)$, where $X \in \mathbb{F}_2^{128}$.
4. $R(X)$ represents $\mathsf{AC} \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}(X)$, where $X \in \mathbb{F}_2^{128}$.
5. $A^r(X)$ represents r times of the application of the function A to X.
6. $R^r(X)$ represents r times of the application of the function R to X.
7. $\mathsf{S}(x)$ represents the output of the S-box of AES when the input is $x \in \mathbb{F}_2^8$.
8. $x \cdot y$ represents the product of $x \in \mathbb{F}_2^8$ and $y \in \mathbb{F}_2^8$ where the operation $\cdot$ works in the field $GF(2^8)$ as in MC of AES.

In addition, we also introduce some related notations to describe the AES state. Specifically, the AES state is organized as a 4×4 two-dimensional array. If the AES state is denoted by s, then $s[i][j]$ represents the byte located in the i-th row and j-th column, as shown in Figure 1. Moreover, to group several bytes of the AES state, we introduce the following notations:

$$\begin{aligned} s[Diag(i)] &= \{s[0][i], s[1][i+1], s[2][i+2], s[3][i+3]\}, \\ s[Col(i)] &= \{s[0][i], s[1][i], s[2][i], s[3][i]\}, \\ s[shiftCol(i)] &= \{s[0][i], s[1][i-1], s[2][i-2], s[3][i-3]\}, \end{aligned}$$

where the indices are considered within modulo 4. For simplicity, we use $s[Col(i_0, i_1, i_2)]$ to represent to state words $s[Col(i_0)] \bigcup s[Col(i_1)] \bigcup s[Col(i_2)]$. Similarly,

$$s[shiftCol(i_0, i_1, i_2)] = s[shiftCol(i_0)] \bigcup s[shiftCol(i_1)] \bigcup s[shiftCol(i_2)].$$

[0][0]	[0][1]	[0][2]	[0][3]
[1][0]	[1][1]	[1][2]	[1][3]
[2][0]	[2][1]	[2][2]	[2][3]
[3][0]	[3][1]	[3][2]	[3][3]

Figure 1: The AES state

2.2 The Initialization Phase of AEGIS-128

AEGIS-128 is an authenticated encryption (AE) scheme composed of four phases: initialization, processing the associated data, encryption and finalization. This works only focuses on the initialization phase.

The state of AEGIS-128 consists of five 128-bit words. For simplicity, we denote the initial state by $(X_0^0, X_0^1, X_0^2, X_0^3, X_0^4)$. Similar to block ciphers, there is a round update function to update the AEGIS-128 state and we denote the state after r rounds by $(X_r^0, X_r^1, X_r^2, X_r^3, X_r^4)$. The round update function is depicted in Figure 2.

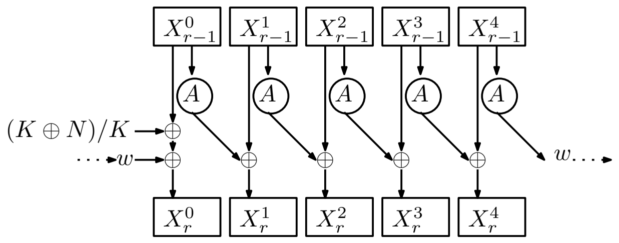

Figure 2: The illustration of the round update function of AEGIS-128

The inputs of the initialization phase are composed of a 128-bit nonce N and a 128-bit key K. There are also two 128-bit constants (C_0, C_1) defined in AEGIS-128. First, the state is initialized in the following way:

$$X_0^0 = K \oplus N, X_0^1 = C_0, X_0^2 = C_1, X_0^3 = K \oplus C_0, X_0^4 = K \oplus C_1.$$

Then, the state will be updated for 10 rounds, i.e. computed until $(X_{10}^0, \ldots, X_{10}^4)$. When $r \geq 1$ is odd, X_r^0 is updated as follows:

$$X_r^0 = X_{r-1}^0 \oplus A(X_{r-1}^4) \oplus K.$$

When $r > 0$ is even, X_r^0 is updated as follows:

$$X_r^0 = X_{r-1}^0 \oplus A(X_{r-1}^4) \oplus K \oplus N.$$

For the remaining state words, they are updated in the same way in each round:

$$X_r^{i+1} = X_{r-1}^{i+1} \oplus A(X_{r-1}^i),$$

where $1 \leq r \leq 10$ and $0 \leq i \leq 3$.

When the associated data is empty, from the ciphertext, the attacker is able to know the output

$$X_{10}^1 \oplus X_{10}^4 \oplus X_{10}^2 \wedge X_{10}^3.$$

When the initialization phase is reduced to r rounds, we denote the output by θ, as specified below:

$$\theta = X_r^1 \oplus X_r^4 \oplus X_r^2 \wedge X_r^3.$$

2.3 The Initialization Phase of Tiaoxin

The Tiaoxin state is composed of thirteen 128-bit words. For convenience, we denote the initial state of Tiaoxin by

$$(U_0^0, U_0^1, U_0^2, W_0^0, W_0^1, W_0^2, W_0^3, Y_0^0, Y_0^1, Y_0^2, Y_0^3, Y_0^4, Y_0^5).$$

Similarly, the inputs of the initialization phase consist of a 128-bit nonce N and a 128-bit key K. There are also two 128-bit constants (Z_0, Z_1) defined in Tiaoxin. First, the initial state is filled in the following way:

$$\begin{aligned}
&U_0^0 = K, U_0^1 = K, U_0^2 = N,\\
&W_0^0 = K, W_0^1 = K, W_0^2 = N, W_0^3 = Z_0,\\
&Y_0^0 = K, Y_0^1 = K, Y_0^2 = N, Y_0^3 = Z_1, Y_0^4 = 0, Y_0^5 = 0.
\end{aligned}$$

Then, a same round update function will be iterated for 15 rounds. Denote the state after r rounds of update by

$$(U_r^0, U_r^1, U_r^2, W_r^0, W_r^1, W_r^2, W_r^3, Y_r^0, Y_r^1, Y_r^2, Y_r^3, Y_r^4, Y_r^5).$$

Then, the round function can be formalized as follows:

$$\begin{aligned}
U_{r+1}^0 &= U_r^0 \oplus Z_0 \oplus A(U_r^2)\\
U_{r+1}^1 &= A(U_r^0)\\
U_{r+1}^2 &= U_r^1\\
W_{r+1}^0 &= W_r^0 \oplus Z_1 \oplus A(W_r^3)\\
W_{r+1}^1 &= A(W_r^0)\\
W_{r+1}^i &= W_r^{i-1}\ (i \in \{2,3\})\\
Y_{r+1}^0 &= Y_r^0 \oplus Z_0 \oplus A(Y_r^5)\\
Y_{r+1}^1 &= A(Y_r^0)\\
Y_{r+1}^i &= Y_r^{i-1}\ (i \in \{2,3,4,5\})
\end{aligned}$$

The corresponding illustration can be referred to Figure 3.

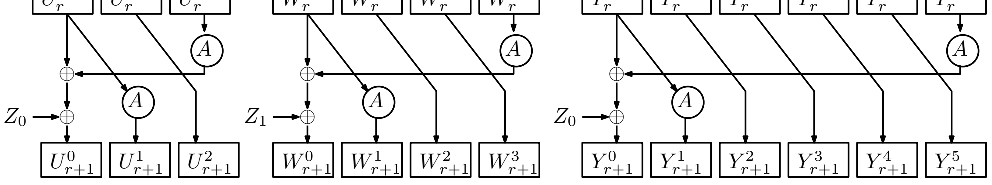

Figure 3: The illustration of the round update function of Tiaoxin

When the associated data is empty and the initialization phase is reduced to r rounds, as the earliest ciphertext is outputted only after one more round update function, if the first two 128-bit message words are 0, the known output (μ_0, μ_1) can be specified as follows:

$$\begin{aligned}
\mu_0 &= U_{r+1}^0 \oplus U_{r+1}^2 \oplus W_{r+1}^1 \oplus Y_{r+1}^3 \wedge W_{r+1}^3,\\
\mu_1 &= Y_{r+1}^0 \oplus W_{r+1}^2 \oplus U_{r+1}^1 \oplus Y_{r+1}^5 \wedge U_{r+1}^2.
\end{aligned}$$

Therefore, it is equivalent to that there are 16 rounds for the initialization phase of Tiaoxin.

3 Integral Properties for the AES Round Function

In the proposal of AES, the designers presented an efficient integral distinguisher for 3-round AES [DR02] and it is still the basis of the best key-recovery attack on 6-round AES in the single key setting. For completeness, the 3-round integral distinguisher is depicted in Figure 4, where $\mathcal{A}$ denotes that the corresponding byte takes all 2^8 possible values, $\mathcal{C}$ denotes that the corresponding byte takes a constant value and $\mathcal{B}$ denotes that the corresponding byte is balanced, i.e. its sum is zero. Formally, the following relation holds:

$$\sum_{s[0][0]\in\mathbb{F}_2^8} R^3(s) = 0.$$

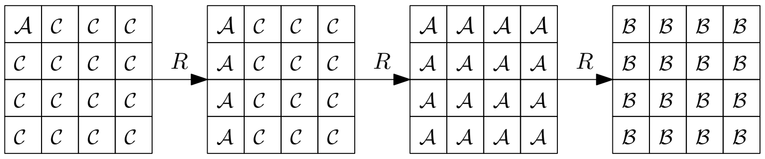

Figure 4: The integral distinguisher for 3-round AES

It is well-known that the 3-round distinguisher can be trivially extended to a 4-round one. Specifically, if $s[Diag(0)]$ traverses all 2^{32} possible values and the remaining bytes of s are assigned to random constants, the sum of all the outputs after 4-round AES must be zero, i.e.

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}} R^4(s) = 0. \tag{1}$$

It is simple to explain why the above equation holds. Specifically, for such an input set of s, the first column of $R(s)$ must also take all 2^{32} possible values and the remaining columns are still constants. Therefore, the set of $R(s)$ can be divided into 2^{24} different subsets according to the value of $(R(s)[1][0], R(s)[2][0], R(s)[3][0])$. Obviously, for each subset, if it passes through 3-round AES further, the sum of the outputs must be zero according to the 3-round integral distinguisher. As the sum of the outputs for each subset is zero, the sum of all the outputs must be zero.

3.1 New Integral Properties for the AES Round Function

Due to the special structure of AEGIS and Tiaoxin, some complex integral properties will occur while they will never occur in real AES. We have to emphasize that the proof of these properties is non-intuitive as we will be faced with a multiset of values which cannot be accurately captured by $\mathcal{A}$, $\mathcal{C}$ or $\mathcal{B}$. Specifically, the multiset has the feature that the same value will appear an even number of times. Indeed, such a feature has once been utilized to break the SASAS scheme in whitebox cryptography [BS01].

Property 1. *Given 3 arbitrary 128-bit constant values c_0, c_1 and c_2, for an arbitrary function $f : \mathbb{F}_2^{128} \to \mathbb{F}_2^l$ where l is an arbitrary positive integer, and for an arbitrary choice of (i, j) $(0 \le i, j \le 3)$, the following property must hold:*

$$\sum_{s[i][j]\in\mathbb{F}_2^8} f(R(s \oplus c_0) \oplus R(s \oplus c_1) \oplus c_2) = 0. \tag{2}$$

Proof. For better understanding, the computation of $f(R(s \oplus c_0) \oplus R(s \oplus c_1) \oplus c_2)$ is illustrated in Figure 5. Due to the symmetry of the AES round function, we only need to

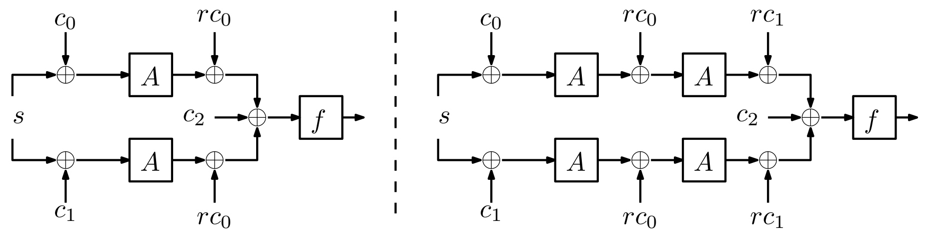

Figure 5: Illustration of the studied functions, where rc_0 and rc_1 are round constants. The left one is for Property 1 and the right one is for Property 3.

prove[2]

$$\sum_{s[0][0]\in\mathbb{F}_2^8} f(R(s\oplus c_0)\oplus R(s\oplus c_1)\oplus c_2)=0.$$

The remaining cases can be proved in a similar way.

Let $s_0=\mathsf{SB}(s\oplus c_0)$ and $s_1=\mathsf{SB}(s\oplus c_1)$. Then, we have

$$R(s\oplus c_0)\oplus R(s\oplus c_1) \quad = \quad \mathsf{MC}\circ\mathsf{SR}(s_0\oplus s_1).$$

As s only varies at $s[0][0]$, we denote the value of s_i by s_i^α $(0\leq i\leq 1)$ when $s[0][0]=\alpha$. When $c_0[0][0]=c_1[0][0]$, $s_0\oplus s_1$ is constant and hence $R(s\oplus c_0)\oplus R(s\oplus c_1)$ is constant. In this case,

$$\sum_{s[0][0]\in\mathbb{F}_2^8} f(R(s\oplus c_0)\oplus R(s\oplus c_1)\oplus c_2)=0.$$

When $c_0[0][0]\neq c_1[0][0]$, we have

$$s_0^\alpha\oplus s_1^\alpha \quad = \quad s_0^{\alpha\oplus c_0[0][0]\oplus c_1[0][0]}\oplus s_1^{\alpha\oplus c_0[0][0]\oplus c_1[0][0]}.$$

In other words, when $s[0][0]$ traverses all the 2^8 possible values, the same value of $s_0\oplus s_1$ will appear an even number of times. Therefore, the same value of $\mathsf{MC}\circ\mathsf{SR}(s_0\oplus s_1)$ must appear an even number of times and hence

$$\sum_{s[0][0]\in\mathbb{F}_2^8} f(R(s\oplus c_0)\oplus R(s\oplus c_1)\oplus c_2)=0.$$

This completes the proof. □

Based on Property 1, there will be a special integral distinguisher for $n+1$ rounds of the AES round function, i.e. for an arbitrary choice of (i,j) $(0\leq i,j\leq 3)$, we have

$$\sum_{s[i][j]\in\mathbb{F}_2^8} R^n(R(s\oplus c_0)\oplus R(s\oplus c_1)\oplus c_2)=0.$$

However, such an integral distinguisher will never appear in real AES.

Property 2. *Given 3 arbitrary 128-bit values c_0, c_1 and c_2, for an arbitrary choice of (i,j) $(0\leq i,j\leq 3)$, the following property must hold:*

$$\sum_{s[i][j]\in\mathbb{F}_2^8} R(R(R(s)\oplus c_0)\oplus R(R(s)\oplus c_1)\oplus c_2)=0. \qquad (3)$$

[2]Indeed, from our proof, it can be found that we indeed only need to prove $\sum_{s[0][0]\in\mathbb{F}_2^8} f(R(s)\oplus R(s\oplus c_1))=0$.

Proof. The basic idea to prove this property is similar to that used in the proof of Property 1. Similarly, we only need to focus on the proof of

$$\sum_{s[0][0]\in\mathbb{F}_2^8} R(R(R(s)\oplus c_0)\oplus R(R(s)\oplus c_1)\oplus c_2)=0.$$

Let $t=R(s)$ and then our aim is to study the property of $R(t\oplus c_0)\oplus R(t\oplus c_1)$. As $s[0][0]=\mathcal{A}$ and the remaining bytes of s are all $\mathcal{C}$, there will be

$$\begin{aligned}&t[0][0]=\mathcal{A},t[1][0]=\mathcal{A},t[2][0]=\mathcal{A},t[3][0]=\mathcal{A},\\&t[i][j]=\mathcal{C}\quad(0\le i\le 3,1\le j\le 3).\end{aligned}$$

Let

$$u=R(t\oplus c_0)\oplus R(t\oplus c_1).$$

For the first column of u, it can be deduced that

$$\begin{aligned}u[0][0]&=2\cdot\mathsf{S}(t[0][0]\oplus c_0[0][0])\oplus 2\cdot\mathsf{S}(t[0][0]\oplus c_1[0][0])\oplus\epsilon_{0,0},\\u[1][0]&=\mathsf{S}(t[0][0]\oplus c_0[0][0])\oplus\mathsf{S}(t[0][0]\oplus c_1[0][0])\oplus\epsilon_{1,0},\\u[2][0]&=\mathsf{S}(t[0][0]\oplus c_0[0][0])\oplus\mathsf{S}(t[0][0]\oplus c_1[0][0])\oplus\epsilon_{2,0},\\u[3][0]&=3\cdot\mathsf{S}(t[0][0]\oplus c_0[0][0])\oplus 3\cdot\mathsf{S}(t[0][0]\oplus c_1[0][0])\oplus\epsilon_{3,0},\end{aligned}$$

where $\epsilon_{i,j}$ are constants depending on (c_0,c_1) and the constant part of t.

If denoting the value of u by u^α when $t[0][0]=\alpha$, there must be

$$\begin{aligned}u^\alpha[0][0]&=u^{\alpha\oplus c_0[0][0]\oplus c_1[0][0]}[0][0],\\u^\alpha[1][0]&=u^{\alpha\oplus c_0[0][0]\oplus c_1[0][0]}[1][0],\\u^\alpha[2][0]&=u^{\alpha\oplus c_0[0][0]\oplus c_1[0][0]}[2][0],\\u^\alpha[3][0]&=u^{\alpha\oplus c_0[0][0]\oplus c_1[0][0]}[3][0].\end{aligned}$$

In other words, if $c_0[0][0]=c_1[0][0]$, $u[Col(0)]$ will be constant. If $c_0[0][0]\neq c_1[0][0]$, the same value of $u[Col(0)]$ will appear an even number of times.

Similarly, we can write the expressions for bytes of u located in the remaining three columns. Due to the symmetry of the AES round function, the same conclusion can be derived. Specifically, if $c_0[i][0]=c_1[i][0]$, $u[Col(4-i)]$ will be constant. If $c_0[i][0]\neq c_1[i][0]$, the same value of $u[Col(4-i)]$ will appear an even number of times, where the indices are considered within modulo 4. Therefore, it can be derived that either $\mathsf{S}(u[i][j]\oplus c_2[i][j])$ is $\mathcal{C}$ or the same value of $\mathsf{S}(u[i][j]\oplus c_2[i][j])$ must appear an even number of times. For both cases, there must be

$$\sum_{s[0][0]\in\mathbb{F}_2^8}\mathsf{SB}\circ(R(R(s)\oplus c_0)\oplus R(R(s)\oplus c_1)\oplus c_2)\quad=\quad 0.$$

As both SR and MC are linear operations, we have

$$\sum_{s[0][0]\in\mathbb{F}_2^8}R(R(R(s)\oplus c_0)\oplus R(R(s)\oplus c_1)\oplus c_2)\quad=\quad 0,$$

which completes the proof. □

The difference between Property 1 and Property 2 should be emphasized. Specifically, in the proof of Property 1, we view the whole AES state as a unit and we derive that the same value of the AES state will appear an even number of times. However, in the proof of

Property 2, we view each byte of the AES state as a unit and derive that the same value of each byte will appear even times. It is not difficult to derive that we lose the integral property for

$$\sum_{s[0][0]\in\mathbb{F}_2^8}\mathsf{SB}\circ R(R(R(s)\oplus c_0)\oplus R(R(s)\oplus c_1)\oplus c_2).$$

Property 3. *Given 3 arbitrary 128-bit values c_0, c_1 and c_2, for an arbitrary function $f:\mathbb{F}_2^{128}\rightarrow\mathbb{F}_2^l$ where l is an arbitrary positive integer, and for an arbitrary i satisfying $0\leq i\leq 3$, the following property must hold:*

$$\sum_{s[Diag(i)]\in\mathbb{F}_2^{32}}f(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus c_2)=0. \tag{4}$$

Proof. The computation of the function $f(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus c_2)$ is depicted in Figure 5. Due to the symmetry of the AES round function, we only need to focus on the proof of

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}}f(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus c_2)=0.$$

Let $p_0=\mathsf{SB}\circ R(s\oplus c_0)$ and $p_1=\mathsf{SB}\circ R(s\oplus c_1)$. In this way, we have

$$R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\quad=\quad\mathsf{MC}\circ\mathsf{SR}(p_0\oplus p_1).$$

When only $s[Diag(0)]$ varies, we have that only $p_0[Col(0)]$ and $p_1[Col(0)]$ will vary while the remaining bytes of them are constant. When $s[Diag(0)]$ takes the value $(\alpha_0,\alpha_1,\alpha_2,\alpha_3)$, denote the value of p_i by $p_i^{\alpha_0,\alpha_1,\alpha_2,\alpha_3}$ $(0\leq i\leq 1)$.

When $c_0[Diag(0)]=c_1[Diag(0)]$, it can be deduced that $p_0\oplus p_1$ is constant. Therefore, when only $s[Diag(0)]$ varies, $R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)$ is constant and we have

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}}f(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus c_2)=0.$$

Let $\delta[i][j]=c_0[i][j]\oplus c_1[i][j]$, when $c_0[Diag(0)]\neq c_1[Diag(0)]$, we can derive that

$$\begin{aligned}&p_0^{\alpha_0,\alpha_1,\alpha_2,\alpha_3}\oplus p_1^{\alpha_0,\alpha_1,\alpha_2,\alpha_3}\\=\;&p_0^{\alpha_0\oplus\delta[0][0],\alpha_1\oplus\delta[1][1],\alpha_2\oplus\delta[2][2],\alpha_3\oplus\delta[3][3]}\oplus p_1^{\alpha_0\oplus\delta[0][0],\alpha_1\oplus\delta[1][1],\alpha_2\oplus\delta[2][2],\alpha_3\oplus\delta[3][3]}.\end{aligned}$$

In other words, when $s[Diag(0)]$ traverses all the 2^{32} possible values, the same value of $p_0\oplus p_1$ will appear an even number of times. This is equivalent to say that the same value of $R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)$ will appear an even number of times. As a result, we also have

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}}f(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus c_2)=0,$$

which completes the proof. □

According to Property 3, there exists an integral property for $n+2$ rounds of the AES round function, i.e. for an arbitrary i $(0\leq i\leq 3)$, we have

$$\sum_{(s[Diag(i)])\in\mathbb{F}_2^{32}}R^n(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus c_2)=0.$$

The conditional integral property. However, for an arbitrary choice of the four 128-bit values of (c_0, c_1, c_2, c_3), there is no deterministic property for the sum

$$\sum_{s[Diag(i)]\in\mathbb{F}_2^{32}} R^2(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus R^2(s\oplus c_2)\oplus c_3).$$

To obtain a deterministic property, some additional conditions can be added. Specifically, from the proof of Property 3, when $c_0[Diag(i)] = c_1[Diag(i)]$ or $c_0[Diag(i)] = c_2[Diag(i)]$ or $c_1[Diag(i)] = c_2[Diag(i)]$, combined with the integral distinguisher for 4-round AES, there must be

$$\sum_{s[Diag(i)]\in\mathbb{F}_2^{32}} R^2(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1)\oplus R^2(s\oplus c_2)\oplus c_3) = 0. \tag{5}$$

Apart from the above properties, it is necessary to prove some additional integral properties to increase the accuracy of our integral distinguishers.

Property 4. *Given an arbitrary 128-bit value c_0, for any i satisfying $0 \le i \le 3$, the following property must hold:*

$$\sum_{s[Col(i)]\in\mathbb{F}_2^{32}} R(s)\wedge R(R(s)\oplus c_0) = 0. \tag{6}$$

Proof. Due to the symmetry of the AES round function, as in all the above proofs, we only need to prove

$$\sum_{s[Col(0)]\in\mathbb{F}_2^{32}} R(s)\wedge R(R(s)\oplus c_0) = 0.$$

Firstly, we focus on the first three columns of $R(s)\wedge R(R(s)\oplus c_0)$. As $s[Col(0)]$ takes all the 2^{32} possible values, the set of s can be divided into 2^{24} different subsets according to $(s[0][0], s[2][0], s[3][0])$. In this way, for each subset of s, $R(s)[Col(0,1,2)]$ will be a fixed constant. For each such subset of s, each byte of $R(R(s)\oplus c_0)$ will take all the 2^8 possible values. Therefore, for each such subset, we have

$$\sum_{s[1][0]\in\mathbb{F}_2^{8}} (R(s)\wedge R(R(s)\oplus c_0))[Col(0,1,2)] = 0.$$

Similarly, it can be simply derived that

$$\sum_{s[0][0]\in\mathbb{F}_2^{8}} (R(s)\wedge R(R(s)\oplus c_0))[Col(1,2,3)] = 0.$$

Consequently,

$$\sum_{s[Col(0)]\in\mathbb{F}_2^{32}} R(s)\wedge R(R(s)\oplus c_0) = 0,$$

which completes the proof. □

Property 5. *Given an arbitrary 128-bit value c_0, for any i satisfying $0 \le i \le 3$, the following property must hold:*

$$\sum_{s[Diag(i)]\in\mathbb{F}_2^{32}} R(R^2(s)\oplus s\oplus c_0) = 0. \tag{7}$$

Proof. As in all the above proofs, it is sufficient to prove

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}} R(R^2(s)\oplus s\oplus c_0)=0.$$

Consider the case when only $s[0][0]$ takes all the 2^8 possible values while the remaining bytes of s are constants. For such a set of inputs, except $(R^2(s)\oplus s)[0][0]$, each byte of $R^2(s)\oplus s$ will independently take all the 2^8 possible values. In other words, when $s[Diag(0)]$ takes all the 2^{32} possible values, the same value of $(R^2(s)\oplus s)[i][j]$ with $(i,j)\neq(0,0)$ will appear an even number of times as there are 2^{24} different values of $(s[1][1],s[2][2],s[3][3])$ and 2^{24} is an even number.

From a different perspective, consider the set of inputs where only $s[1][1]$ takes all the 2^8 possible values. In this case, except $(R^2(s)\oplus s)[1][1]$, each byte of $R^2(s)\oplus s$ will independently take all the 2^8 possible values. Based on similar reasons, the same value of $(R^2(s)\oplus s)[i][j]$ with $(i,j)\neq(1,1)$ will appear an even number of times when $s[Diag(0)]$ traverses all the 2^{32} possible values.

Combining both cases, when $s[Diag(0)]$ takes all the 2^{32} possible values, the same value of each byte of $(R^2(s)\oplus s)$ will appear an even number of times. However, it should be emphasized that the same value of $R^2(s)\oplus s$ will not necessarily appear an even number of times. As a result, we have

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}} R(R^2(s)\oplus s\oplus c_0)=0,$$

which completes the proof. □

4 Cryptanalysis of 5-Round AEGIS-128

In this section, we will describe how to identify the weak keys in 5-round AEGIS-128 by tracing the expressions of the internal states in terms of the initial state. As the AEGIS-128 state is composed of five 128-bit words and the round update function treats each word as a unit, we are motivated to write the expressions of the internal states in terms of the initial state by treating a 128-bit word rather than 1 bit as a unit. Therefore, it is no more difficult to write the accurate expressions, while it is almost impossible in bit level.

4.1 Writing the Expressions of 5-Round AEGIS-128

To understand how the weak keys influence the expressions, it is essential to know the original expressions for an arbitrary key. For simplicity, when writing the expressions, we omit the constants and only focus on how the nonce evolves as the state is updated. Consequently, in the following, $A(N)$ may represent $A(N\oplus c)$ where c is a constant depending on the key and the constant part of the initial state. In addition, when the state word does not depend on the nonce N, it is simply written as 0. This way will make the expressions more explicit and readable.

The initial state of AEGIS-128 is defined as below:

$$X_0^0=K\oplus N, X_0^1=C_0, X_0^2=C_1, X_0^3=K\oplus C_0, X_0^4=K\oplus C_1.$$

Therefore, the expressions of $(X_1^0,\ldots,X_1^4)$ can be written as follows:

$$X_1^0=N, X_1^1=A(N), X_1^2=0, X_1^3=0, X_1^4=0.$$

Similarly, we can write the expressions of $(X_r^0,\ldots,X_r^4)$ for $2\leq r\leq 5$.

When $r = 2$, we have

$$X_2^0 = 0, X_2^1 = A(N) \oplus A(N), X_2^2 = A(A(N)), X_2^3 = 0, X_2^4 = 0.$$

When $r = 3$, the expressions are

$$\begin{aligned}&X_3^0 = 0, X_3^1 = A(N) \oplus A(N), X_3^2 = A(A(N)) \oplus A(A(N) \oplus A(N)),\\ &X_3^3 = A(A(A(N))), X_3^4 = 0.\end{aligned}$$

When $r = 4$, there will be

$$\begin{aligned}X_4^0 &= N,\\ X_4^1 &= A(N) \oplus A(N),\\ X_4^2 &= A(A(N)) \oplus A(A(N) \oplus A(N)) \oplus A(A(N) \oplus A(N)),\\ X_4^3 &= A(A(A(N))) \oplus A(A(A(N)) \oplus A(A(N) \oplus A(N))),\\ X_4^4 &= A(A(A(A(N)))).\end{aligned}$$

Finally, when $r = 5$, we have

$$\begin{aligned}X_5^0 &= N \oplus A(A(A(A(A(N))))),\\ X_5^1 &= A(N) \oplus A(N) \oplus A(N),\\ X_5^2 &= A(A(N)) \oplus A(A(N) \oplus A(N)) \oplus A(A(N) \oplus A(N)) \oplus A(A(N) \oplus A(N)),\\ X_5^3 &= A(A(A(N))) \oplus A(A(A(N)) \oplus A(A(N) \oplus A(N)))\\ &\quad \oplus A(A(A(N)) \oplus A(A(N) \oplus A(N)) \oplus A(A(N) \oplus A(N))),\\ X_5^4 &= A(A(A(A(N)))) \oplus A(A(A(A(N))) \oplus A(A(A(N)) \oplus A(A(N) \oplus A(N)))).\end{aligned}$$

According to the output of AEGIS-128, if targeting 5 AEGIS-128 initialization rounds, the attacker can only know

$$\theta = X_5^1 \oplus X_5^4 \oplus X_5^2 \wedge X_5^3. \tag{8}$$

It seems that the simple integral distinguisher for 4-round AES can be directly applied to 5-round AEGIS-128 as X_5^0 does not influence the output. However, the logic AND operation between X_5^2 and X_5^3 simply makes it impossible. Therefore, we are motivated to investigate whether it is possible to simplify the expressions of X_5^2 and X_5^3 by adding proper conditions. Indeed, similar ideas are commonly used in symmetric-key cryptanalysis, which is to add additional conditions to slow down the propagation of variables, although most of them are deduced by carefully tracing the influence of a certain bit condition. However, it seems difficult to analyze the constructions based on AES from the bit level. Thus, we will study how to add conditions from the byte level as it is more compatible with the AES specification.

4.2 Adding Conditions To Simplify the Expressions

To carefully investigate how the conditions affect the expressions, it is necessary to write the accurate expressions of the internal states in terms of the nonce, the key and the constant part of the initial state. In the following, we will expand on how the conditions are derived and how the expressions are simplified.

Similarly, the initial state is defined as below:

$$X_0^0 = K \oplus N, X_0^1 = C_0, X_0^2 = C_1, X_0^3 = K \oplus C_0, X_0^4 = K \oplus C_1.$$

Our aim is to write the expressions of $(X_r^0, \ldots, X_r^4)$ for $1 \leq r \leq 5$ by involving all the information. In the process, we will continuously introduce new variables C_i $(i > 1)$ to represent constant values to reduce the length of the expressions. To save space, we will not repeat the definitions of these new variables.

When $r = 1$, we have

$$\begin{aligned}
X_1^0 &= N \oplus A(K \oplus C_1)\\
X_1^1 &= C_0 \oplus A(K \oplus N)\\
X_1^2 &= C_1 \oplus A(C_0) = C_2\\
X_1^3 &= K \oplus C_0 \oplus A(C_1) = K \oplus C_3\\
X_1^4 &= K \oplus C_1 \oplus A(K \oplus C_0).
\end{aligned}$$

When $r = 2$, we have

$$\begin{aligned}
X_2^0 &= K \oplus A(K \oplus C_1) \oplus A(K \oplus C_1 \oplus A(K \oplus C_0)) = C_4\\
X_2^1 &= C_0 \oplus A(K \oplus N) \oplus A(N \oplus A(K \oplus C_1))\\
X_2^2 &= C_2 \oplus A(C_0 \oplus A(K \oplus N))\\
X_2^3 &= K \oplus C_3 \oplus A(C_2) = C_5\\
X_2^4 &= K \oplus C_1 \oplus A(K \oplus C_0) \oplus A(K \oplus C_3) = C_6.
\end{aligned}$$

When $r = 3$, we have

$$\begin{aligned}
X_3^0 &= K \oplus C_4 \oplus A(C_6) = C_7\\
X_3^1 &= C_0 \oplus A(K \oplus N) \oplus A(N \oplus A(K \oplus C_1)) \oplus A(C_4)\\
X_3^2 &= C_2 \oplus A(C_0 \oplus A(K \oplus N)) \oplus A(C_0 \oplus A(K \oplus N) \oplus A(N \oplus A(K \oplus C_1)))\\
X_3^3 &= C_5 \oplus A(C_2 \oplus A(C_0 \oplus A(K \oplus N)))\\
X_3^4 &= C_6 \oplus A(C_5) = C_8.
\end{aligned}$$

From the expressions of X_3^1 and X_3^2, it can be found that both of them contain the expression $A(K \oplus N) \oplus A(N \oplus A(K \oplus C_1))$. From the proof of Property 3, when the following condition holds:

$$A(K \oplus C_1)[Diag(0)] \quad = \quad K[Diag(0)], \tag{9}$$

the expression $A(K \oplus N) \oplus A(N \oplus A(K \oplus C_1))$ will take a constant value if only $N[Diag(0)]$ varies. Therefore, in the following, we only consider the expressions when $N[Diag(0)]$ takes all the 2^{32} possible values while the remaining bytes of N take random constant values. In this case, when Equation 9 holds, $A(K \oplus N) \oplus A(N \oplus A(K \oplus C_1))$ is constant. Hence, we define that

$$A(K \oplus N) \oplus A(N \oplus A(K \oplus C_1)) \oplus C_0 = C_9.$$

In this way, the expressions of $(X_3^0, X_3^1, X_3^2, X_3^3, X_3^4)$ can be further simplified, as shown below:

$$\begin{aligned}
X_3^0 &= C_7\\
X_3^1 &= C_9 \oplus A(C_4) = C_{10}\\
X_3^2 &= C_2 \oplus A(C_0 \oplus A(K \oplus N)) \oplus A(C_9) = A(C_0 \oplus A(K \oplus N)) \oplus C_{11}\\
X_3^3 &= C_5 \oplus A(C_2 \oplus A(C_0 \oplus A(K \oplus N)))\\
X_3^4 &= C_8.
\end{aligned}$$

From the simplified expressions, we further observed that we could introduce an intermediate variable T to represent $C_0 \oplus A(K \oplus N)$, i.e.

$$T \quad = \quad C_0 \oplus A(K \oplus N). \tag{10}$$

In this way, the expressions can be further simplified, as shown below:

$$\begin{aligned}X_3^0 &= C_7\\ X_3^1 &= C_{10}\\ X_3^2 &= A(T)\oplus C_{11}\\ X_3^3 &= C_5\oplus A(C_2\oplus A(T))\\ X_3^4 &= C_8.\end{aligned}$$

It can also be found later that introducing an intermediate variable T is crucial to understand the key-recovery attack on 5-round AEGIS-128.

Consequently, when $r=4$, there will be

$$\begin{aligned}X_4^0 &= K\oplus N\oplus C_7\oplus A(C_8) = N\oplus C_{12}\\ X_4^1 &= C_{10}\oplus A(C_7) = C_{13}\\ X_4^2 &= A(T)\oplus C_{11}\oplus A(C_{10}) = A(T)\oplus C_{14}\\ X_4^3 &= A(A(T)\oplus C_2)\oplus C_5\oplus A(A(T)\oplus C_{11})\\ X_4^4 &= C_8\oplus A(A(A(T)\oplus C_2)\oplus C_5).\end{aligned}$$

Finally, we consider the case $r=5$ and there will be

$$\begin{aligned}X_5^0 &= K\oplus N\oplus C_{12}\oplus A(C_8\oplus A(A(A(T)\oplus C_2)\oplus C_5))\\ X_5^1 &= C_{13}\oplus A(N\oplus C_{12})\\ X_5^2 &= A(T)\oplus C_{14}\oplus A(C_{13})\\ X_5^3 &= A(A(T)\oplus C_2)\oplus C_5\oplus A(A(T)\oplus C_{11})\oplus A(A(T)\oplus C_{14})\\ X_5^4 &= C_8\oplus A(A(A(T)\oplus C_2)\oplus C_5)\oplus A(A(A(T)\oplus C_2)\oplus C_5\oplus A(A(T)\oplus C_{11}))\end{aligned}$$

For the output θ, we have

$$\theta = X_5^1\oplus X_5^4\oplus X_5^2\wedge X_5^3. \tag{11}$$

It should be emphasized that the values of T and N are related according to Equation 10. As $N[Diag(0)]$ takes all the 2^{32} possible values and the remaining bytes of N are constants, $T[Col(0)]$ will also traverse all the 2^{32} possible values while the remaining bytes of T are constants. As a result, the values of T can be further divided into 2^{24} different subsets according to the value of $(T[1][0],T[2][0],T[3][0])$, though how to divide them depends on the secret key.

Different from the expression of X_5^1, the expressions of X_5^2 and X_5^3 do not contain the variable N after introducing the variable T. Therefore, there is no need to relate T and N when studying the integral property of $X_5^2\wedge X_5^3$. In other words, we simply treat T as irrelevant to N and $T[Col(0)]$ will take all the 2^{32} possible values. Therefore, we can focus on the case when only $T[0][0]$ is $\mathcal{A}$ while the remaining bytes of T are $\mathcal{C}$. In this case, $X_5^2[Col(1,2,3)]$ are constants, thus resulting that the integral property of $(X_5^2\wedge X_5^3)[Col(1,2,3)]$ indeed only depends on the integral property of $X_5^3[Col(1,2,3)]$. Consequently,

$$\sum_{T[0][0]\in\mathbb{F}_2^8}(X_5^2\wedge X_5^3)[Col(1,2,3)] = 0. \tag{12}$$

Indeed, based on Property 4, there will be

$$\sum_{T[Col(0)]\in\mathbb{F}_2^{32}} X_5^2\wedge X_5^3 = 0. \tag{13}$$

For X_5^1, it is simple to derive that

$$\sum_{T[0][0]\in\mathbb{F}_2^8} X_5^1[Col(1,2,3)] = 0.$$

However, if considering the inverse of the AES round function, we can find that when only $T[0][0]$ is $\mathcal{A}$, $N[i][i]$ $(0 \leq i \leq 3)$ must also be $\mathcal{A}$. Therefore,

$$\sum_{T[0][0]\in\mathbb{F}_2^8} X_5^1 = 0. \tag{14}$$

Finally, we are only left with the integral property of X_5^4. Similarly, as its expression does not contain N, we simply consider the case when only $T[0][0]$ is $\mathcal{A}$ while the remaining bytes of T are $\mathcal{C}$. From the integral property of 3-round AES, we have

$$\sum_{T[0][0]\in\mathbb{F}_2^8} C_8 \oplus A(A(A(T)\oplus C_2)\oplus C_5) = 0.$$

According to Property 2, we have

$$\sum_{T[0][0]\in\mathbb{F}_2^8} A(A(A(T)\oplus C_2)\oplus C_5 \oplus A(A(T)\oplus C_{11})) = 0.$$

Hence, there will be

$$\sum_{T[0][0]\in\mathbb{F}_2^8} X_5^4 = 0. \tag{15}$$

Combining Equation 12, Equation 14 and Equation 15, we thus have

$$\sum_{T[0][0]\in\mathbb{F}_2^8} \theta[Col(1,2,3)] = 0, \tag{16}$$

which will be the basis of our key-recovery attack.

Combining Equation 13, Equation 14 and Equation 15, we directly obtain a distinguisher for 5-round AEGIS-128 with time complexity and data complexity 2^{32}, as shown below:

$$\sum_{N[Diag(0)]\in\mathbb{F}_2^{32}} \theta = 0. \tag{17}$$

4.3 The Key-Recovery Attack on 5-Round AEGIS-128

Based on the above analysis, we can design a weak key recovery attack. First of all, a weak key should satisfy the following condition:

$$A(K\oplus C_1)[Diag(0)] = K[Diag(0)].$$

Notice that after guessing $K[Diag(0)]$, it is feasible to compute $A(K\oplus C_1)[0][0]$ and check whether it is identical to the guessed value of $K[0][0]$. In other words, there will be 2^{32-8} valid values for $K[Diag(0)]$. Therefore, we can first compute and store all the possible 2^{24} values of $K[Diag(0)]$ in a table denoted by KT_0. The reason why there are 2^{24} possible values can be simply explained. When $(K[0][0], K[1][1], K[2][2])$ is fixed and $K[3][3]$ is traversed, $A(K\oplus C_1)[0][0]$ must traverse 2^8 different values and there is only one value which can match the guessed $K[0][0]$. As there are 2^{24} possible values for $(K[0][0], K[1][1], K[2][2])$, there must be 2^{24} different valid values for $K[Diag(0)]$.

After the table KT_0 is constructed, the correct value of $K[Diag(0)]$ can be simply recovered in the following way:

Step 1: Construct a set of size 2^8 for T by traversing $T[0][0]$ while $T[i][j]$ $(i, j) \neq (0, 0)$ is set as a random constant.

Step 2: For each candidate of $K[Diag(0)]$ in KT_0, construct the set of size 2^8 for N. Specifically, compute the 2^8 different values for $N[Diag(0)]$ based on $T = C_0 \oplus A(K \oplus N)$. For $N[Diag(i)]$ $(1 \leq i \leq 3)$, they are set as random constants. Encrypt all possible 2^8 different values of N with 5-round AEGIS-128 and collect the corresponding 2^8 outputs θ. If

$$\sum \theta[Col(1, 2, 3)] = 0, \tag{18}$$

then the current candidate for $K[Diag(0)]$ is the correct value and store it. Otherwise, consider the next candidate for $K[Diag(0)]$ and repeat until all values in KT_0 are traversed.

Complexity evaluation. For a wrong key guess, it is treated as correct with probability 2^{-96}. Hence, we expect that only the correct key will survive. For each candidate for $K[Diag(0)]$, it is necessary to compute 2^8 different N. As there are 2^{24} candidates, the data complexity[3] is upper bounded by 2^{32}. The time complexity is also upper bounded by 2^{32} encryptions.

Experiments. The experiments[4] are performed on the small-scale AES [CMR05]. For both the distinguishing attack and the key-recovery attack, experiments show that they succeed with probability 1. For a random key, the attacks always fail.

Efficiently recovering the weak key. Notice that a weak key satisfies $A(K \oplus C_1)[Diag(0)] = K[Diag(0)]$. After the above procedure, $K[Diag(0)]$ is known. Therefore, there are 96 unknown key bits left. As $K[Diag(0)]$ is known, we can independently guess $K[Diag(i)]$ $(1 \leq i \leq 3)$ and compute $A(K \oplus C_1)[Col(i)]$ and check whether $A(K \oplus C_1)[i][i] = K[i][i]$. Consequently, we can collect 2^{24} candidates for $K[Diag(1)]$, $K[Diag(2)]$ and $K[Diag(3)]$, respectively. In total, there are $2^{24 \times 3} = 2^{72}$ candidates for the 128-bit key. Therefore, the weak key can be recovered with time complexity 2^{72}, which is $2^{96-72} = 2^{24}$ times faster than an exhaustive search. Combining with the procedure to recover $K[Diag(0)]$, the time complexity, data complexity and memory complexity to recover the weak key are 2^{72}, 2^{32} and 2^{25}, respectively.

Remark. When a key is not a weak key, it is expected that in the procedure to recover $K[Diag(0)]$ the event $\theta[Col(1, 2, 3)] = 0$ will not happen during the 2^{24} tests. Indeed, by imposing different conditions $A(K \oplus C_1)[Diag(i)] = K[Diag(i)]$ $(1 \leq i \leq 3)$, we can determine different sets of weak keys and they can be recovered in the similar way. In conclusion, there are 4 different sets of weak keys with 2^{96} keys each. For each set of weak keys, the time complexity and data complexity to recover the correct one are 2^{72} and 2^{32}, respectively.

4.4 Failing in Attacking 5-Round AEGIS-128L

It seems that the above method can be applied to 5-round AEGIS-128L, we are thus motivated to study whether it is actually feasible. However, due to a more clever way to generate the output from the state, the similar attack on 5-round AEGIS-128L cannot work. To explain this, we can carefully study the expressions of the internal states.

[3]It is possible to use fewer data by considering the collision of N. As the current data complexity is already small, we will not address this problem.

[4]See `https://github.com/LFKOKAMI/AEGIS-Tiaoxin-Weak-Keys.git` for the source code.

Similarly, the AEGIS-128L state is composed of eight 128-bit words and we denote the initial state by $(S_0^0, \ldots, S_0^7)$. The inputs of the initialization phase of AEGIS-128L are the same as those in AEGIS-128. According to the specification of AEGIS-128L, the initial state of AEGIS-128L is defined as follows:

$$\begin{aligned}&S_0^0 = K \oplus N, S_0^1 = C_1, S_0^2 = C_0, S_0^3 = C_1,\\ &S_0^4 = K \oplus N, S_0^5 = K \oplus C_0, S_0^6 = K \oplus C_1, S_0^7 = K \oplus C_0.\end{aligned}$$

The round update function is:

$$\begin{aligned}S_{r+1}^0 &= A(S_r^7) \oplus S_r^0 \oplus N,\\ S_{r+1}^4 &= A(S_r^3) \oplus S_r^4 \oplus K,\\ S_{r+1}^j &= A(S_r^{j-1}) \oplus S_r^j \ (j \in \{1,2,3,5,6,7\}),\end{aligned}$$

where $(S_r^0, \ldots, S_r^7)$ denotes the state after r rounds of update.

When writing the expressions of $(S_r^0, \ldots, S_r^7)$ for $1 \leq r \leq 5$, similar to our way to analyze AEGIS-128, we directly introduce new variables P_i $(i \geq 0)$ to represent the constant part of the expression.

When $r = 1$, there will be

$$\begin{aligned}S_1^0 &= A(K \oplus C_0) \oplus K \oplus N \oplus N = P_0,\\ S_1^1 &= A(K \oplus N) \oplus C_1,\\ S_1^2 &= A(C_1) \oplus C_0 = P_1,\\ S_1^3 &= A(C_0) \oplus C_1 = P_2,\\ S_1^4 &= A(C_1) \oplus K \oplus N \oplus K = P_3 \oplus N,\\ S_1^5 &= A(K \oplus N) \oplus K \oplus C_0 = A(K \oplus N) \oplus P_3,\\ S_1^6 &= A(K \oplus C_0) \oplus K \oplus C_1 = P_4,\\ S_1^7 &= A(K \oplus C_1) \oplus K \oplus C_0 = P_5.\end{aligned}$$

When $r = 2$, there will be

$$\begin{aligned}S_2^0 &= A(P_5) \oplus P_0 \oplus N = N \oplus P_6,\\ S_2^1 &= A(P_0) \oplus A(K \oplus N) \oplus C_1 = A(N \oplus K) \oplus P_7,\\ S_2^2 &= A(A(N \oplus K) \oplus C_1) \oplus P_1,\\ S_2^3 &= A(P_1) \oplus P_2 = P_9,\\ S_2^4 &= A(P_2) \oplus P_3 \oplus N \oplus K = N \oplus P_{10},\\ S_2^5 &= A(N \oplus P_3) \oplus A(N \oplus K) \oplus P_3,\\ S_2^6 &= A(A(N \oplus K) \oplus P_3) \oplus P_4,\\ S_2^7 &= A(P_4) \oplus P_5 = P_{11}.\end{aligned}$$

When $r = 3$, there will be

$$\begin{aligned}S_3^0 &= A(P_{11}) \oplus N \oplus P_6 \oplus N = P_{12},\\ S_3^1 &= A(N \oplus P_6) \oplus A(N \oplus K) \oplus P_7,\\ S_3^2 &= A(A(N \oplus K) \oplus P_7) \oplus A(A(N \oplus K) \oplus C_1) \oplus P_1,\\ S_3^3 &= A(A(A(N \oplus K) \oplus C_1) \oplus P_1) \oplus P_9,\\ S_3^4 &= A(P_9) \oplus N \oplus P_{10} \oplus K = N \oplus P_{13},\\ S_3^5 &= A(N \oplus P_{10}) \oplus A(N \oplus P_3) \oplus A(N \oplus K) \oplus P_3,\\ S_3^6 &= A(A(N \oplus P_3) \oplus A(N \oplus K) \oplus P_3) \oplus A(A(N \oplus K) \oplus P_3) \oplus P_4,\end{aligned}$$

$$S_3^7 \quad = \quad A(A(A(N \oplus K) \oplus P_3) \oplus P_4) \oplus P_{11}.$$

As r increases, the expression becomes more and more complex. Thus, we first consider the output of 5-round AEGIS-128L, as shown below:

$$\begin{aligned}\pi_5^0 &= S_5^1 \oplus S_5^6 \oplus S_5^2 \wedge S_5^3,\\ \pi_5^1 &= S_5^2 \oplus S_5^5 \oplus S_5^6 \wedge S_5^7.\end{aligned}$$

It can be found from the expression of S_3^3 that it is impossible to eliminate $A(A(A(N \oplus K) \oplus C_1) \oplus P_1)$ and N has passed through 3 AES rounds. Hence, in the expression of S_5^5, N will pass through 5 AES rounds and it is impossible to reduce the number of AES rounds that N will pass through by adding proper conditions. Similarly, it is impossible to eliminate $A(A(A(N \oplus K) \oplus P_3) \oplus P_4)$ and N also has passed through 3 AES rounds. Therefore, in the expression of S_5^1, N will pass through 5 AES rounds and it is impossible to reduce the number of AES rounds that N will pass through by adding proper conditions. As the conventional integral distinguisher on 3-round AES can not be adapted to 5 rounds, the successful attack on 5-round AEGIS-128 cannot be applied to 5-round AEGIS-128L.

Only for interest, we find that the quadratic parts S_5^2 and S_5^3 can be simplified by adding proper conditions, the expressions of which can be written as follows:

$$\begin{aligned}S_4^1 &= A(P_{12}) \oplus A(N \oplus P_6) \oplus A(N \oplus K) \oplus P_7 = A(N \oplus P_6) \oplus A(N \oplus K) \oplus P_{14},\\ S_4^2 &= A(A(N \oplus P_6) \oplus A(N \oplus K) \oplus P_7)\\ &\quad \oplus A(A(N \oplus K) \oplus P_7) \oplus A(A(N \oplus K) \oplus C_1) \oplus P_1,\\ S_4^3 &= A(A(A(N \oplus K) \oplus P_7) \oplus A(A(N \oplus K) \oplus C_1) \oplus P_1)\\ &\quad \oplus A(A(A(N \oplus K) \oplus C_1) \oplus P_1) \oplus P_9\\ S_5^2 &= A(S_4^1) \oplus S_4^2,\\ S_5^3 &= A(S_4^2) \oplus S_4^3.\end{aligned}$$

Let $H = A(N \oplus K)$. If $P_7[0][0] = C_1[0][0]$, when $H[0][0]$ is $\mathcal{A}$ and remaining 15 bytes of H are all $\mathcal{C}$, it can be known that

$$A(H \oplus P_7) \oplus A(H \oplus C_1)$$

is a constant. Further, if $P_6[Diag(0)] = K[Diag(0)]$, when H takes the pattern as above,

$$A(N \oplus P_6) \oplus A(N \oplus K)$$

is also a constant. Thus, we can specify the conditions that the weak key should satisfy according to P_6 and P_7, as shown below:

$$\begin{aligned}(A(A(K \oplus C_0) \oplus K) \oplus C_1)[0][0] &= C_1[0][0],\\ (A(A(K \oplus C_1) \oplus K \oplus C_0) \oplus A(K \oplus C_0) \oplus K)[Diag(0)] &= K[Diag(0)].\end{aligned}$$

In other words, we have

$$A(A(K \oplus C_0) \oplus K)[0][0] = 0, \tag{19}$$

$$A(A(K \oplus C_1) \oplus K \oplus C_0)[Diag(0)] = A(K \oplus C_0)[Diag(0)]. \tag{20}$$

When the above 40 bit conditions hold and H takes the above input pattern, it is easy to know that S_4^1 and S_4^2 will always be constants. In this way, the expressions can be updated as follows:

$$S_4^1 = P_{15}, S_4^2 = P_{16}, S_4^3 = A(A(A(N \oplus K) \oplus C_1) \oplus P_1) \oplus P_{17}.$$

Thus, we have

$$\begin{aligned} S_5^2 &= A(P_{15}) \oplus P_{16}, \\ S_5^3 &= A(P_{16}) \oplus A(A(H \oplus C_1) \oplus P_1) \oplus P_{17}. \end{aligned}$$

Obviously, the quadratic part is greatly simplified, though we still cannot mount an attack on 5-round AEGIS-128L. However, we believe that revealing how to simplify the quadratic part is important for future research.

5 Cryptanalysis of 8-Round Tiaoxin

From the analysis of AEGIS-128, it can be found that the conditions on the key are used to simplify the quadratic part of the output. In our analysis of Tiaoxin, we will show that the weak keys occur for different reasons. Similarly, our aim is to study the integral property of the expressions of the internal states. Therefore, it is necessary to write the expressions of (U_r^0, U_r^1, U_r^2), $(W_r^0, \ldots, W_r^3)$ and $(Y_r^0, \ldots, Y_r^5)$, respectively, where $1 \leq r \leq 8$.

When $r = 1$, we have

$$U_1^0 = A(N) \oplus Z_0 \oplus K, U_1^1 = A(K), U_1^2 = K$$

$$W_1^0 = A(Z_0) \oplus Z_1 \oplus K, W_1^1 = A(K), W_1^2 = K, W_1^3 = N$$

$$Y_1^0 = A(0) \oplus Z_0 \oplus K, Y_1^1 = A(K), Y_1^2 = K, Y_1^3 = N, Y_1^4 = Z_1, Y_1^5 = 0$$

When $r = 2$, we have

$$\begin{aligned} &U_2^0 = A(K) \oplus Z_0 \oplus A(N) \oplus Z_0 \oplus K = A(N) \oplus A(K) \oplus K, \\ &U_2^1 = A(A(N) \oplus Z_0 \oplus K), U_2^2 = A(K) \end{aligned}$$

$$\begin{aligned} &W_2^0 = A(N) \oplus Z_1 \oplus A(Z_0) \oplus Z_1 \oplus K = A(N) \oplus A(Z_0) \oplus K, \\ &W_2^1 = A(Z_0) \oplus Z_1 \oplus K, W_2^2 = A(K), W_2^3 = K \end{aligned}$$

$$\begin{aligned} &Y_2^0 = A(0) \oplus Z_0 \oplus A(0) \oplus Z_0 \oplus K = K, \\ &Y_2^1 = A(0) \oplus Z_0 \oplus K, Y_2^2 = A(K), Y_2^3 = K, Y_2^4 = N, Y_2^5 = Z_1 \end{aligned}$$

When $r = 3$, we have

$$\begin{aligned} &U_3^0 = A^2(K) \oplus Z_0 \oplus A(N) \oplus A(K) \oplus K = A(N) \oplus Z_2, \\ &U_3^1 = A(A(N) \oplus A(K) \oplus K) = A(A(N) \oplus Z_3), \\ &U_3^2 = A(A(N) \oplus Z_0 \oplus K) = A(A(N) \oplus Z_4) \end{aligned}$$

$$\begin{aligned} &W_3^0 = A(K) \oplus Z_1 \oplus A(N) \oplus A(Z_0) \oplus K = A(N) \oplus Z_5, \\ &W_3^1 = A(A(N) \oplus A(Z_0) \oplus K) = A(A(N) \oplus Z_6), \\ &W_3^2 = A(Z_0) \oplus Z_1 \oplus K = Z_7, W_3^3 = A(K) \end{aligned}$$

$$\begin{aligned} &Y_3^0 = A(Z_1) \oplus Z_0 \oplus K = Z_8, \\ &Y_3^1 = A(K), Y_3^2 = A(0) \oplus Z_0 \oplus K = Z_9, Y_3^3 = A(K), Y_3^4 = K, Y_3^5 = N, \end{aligned}$$

where

$$Z_2 = A^2(K) \oplus Z_0 \oplus A(K) \oplus K, Z_3 = A(K) \oplus K, Z_4 = Z_0 \oplus K,$$

$$Z_5 = A(K) \oplus Z_1 \oplus A(Z_0) \oplus K, Z_6 = A(Z_0) \oplus K, Z_7 = A(Z_0) \oplus Z_1 \oplus K,$$
$$Z_8 = A(Z_1) \oplus Z_0 \oplus K, Z_9 = A(0) \oplus Z_0 \oplus K.$$

When $r = 4$, we have

$$\begin{aligned}
U_4^0 &= A^2(A(N) \oplus Z_4) \oplus Z_0 \oplus A(N) \oplus Z_2 = A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_{10},\\
U_4^1 &= A(A(N) \oplus Z_2),\\
U_4^2 &= A(A(N) \oplus Z_3)\\
\\
W_4^0 &= A^2(K) \oplus Z_1 \oplus A(N) \oplus Z_5 = A(N) \oplus Z_{11},\\
W_4^1 &= A(A(N) \oplus Z_5),\\
W_4^2 &= A(A(N) \oplus Z_6), W_4^3 = Z_7\\
\\
Y_4^0 &= A(N) \oplus Z_0 \oplus Z_8 = A(N) \oplus Z_{12},\\
Y_4^1 &= A(Z_8), Y_4^2 = A(K), Y_4^3 = Z_9, Y_4^4 = A(K), Y_4^5 = K,
\end{aligned}$$

where

$$\begin{aligned}
Z_{10} &= Z_0 \oplus Z_2 = A^2(K) \oplus A(K) \oplus K,\\
Z_{11} &= A^2(K) \oplus Z_1 \oplus Z_5 = A(Z_0) \oplus A^2(K) \oplus A(K) \oplus K,\\
Z_{12} &= Z_0 \oplus Z_8 = A(Z_1) \oplus K.
\end{aligned}$$

When $r = 5$, we have

$$\begin{aligned}
U_5^0 &= A^2(A(N) \oplus Z_3) \oplus Z_0 \oplus A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_{10},\\
&= A^2(A(N) \oplus Z_3) \oplus A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_2\\
U_5^1 &= A(A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_{10}),\\
U_5^2 &= A(A(N) \oplus Z_2)\\
\\
W_5^0 &= A(Z_7) \oplus Z_1 \oplus A(N) \oplus Z_{11} = A(N) \oplus Z_{13},\\
W_5^1 &= A(A(N) \oplus Z_{11}),\\
W_5^2 &= A(A(N) \oplus Z_5), W_5^3 = A(A(N) \oplus Z_6)\\
\\
Y_5^0 &= A(K) \oplus Z_0 \oplus A(N) \oplus Z_{12} = A(N) \oplus Z_{14},\\
Y_5^1 &= A(A(N) \oplus Z_{12}), Y_5^2 = A(Z_8), Y_5^3 = A(K), Y_5^4 = Z_9, Y_5^5 = A(K),
\end{aligned}$$

where

$$Z_{13} = A(Z_7) \oplus Z_1 \oplus Z_{11}, Z_{14} = A(K) \oplus Z_0 \oplus \oplus Z_{12} = Z_8 \oplus A(K).$$

When $r = 6$, we have

$$\begin{aligned}
U_6^0 &= A^2(A(N) \oplus Z_2) \oplus Z_0 \oplus A^2(A(N) \oplus Z_3) \oplus A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_2\\
&= A^2(A(N) \oplus Z_2) \oplus A^2(A(N) \oplus Z_3) \oplus A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_{10}\\
U_6^1 &= A(A^2(A(N) \oplus Z_3) \oplus A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_2),\\
U_6^2 &= A(A^2(A(N) \oplus Z_4) \oplus A(N) \oplus Z_{10})\\
\\
W_6^0 &= A^2(A(N) \oplus Z_6) \oplus Z_1 \oplus A(N) \oplus Z_{13},
\end{aligned}$$

$$\begin{aligned}
W_6^1 &= A(A(N) \oplus Z_{13}), \\
W_6^2 &= A(A(N) \oplus Z_{11}), W_6^3 = A(A(N) \oplus Z_5)
\end{aligned}$$

$$\begin{aligned}
Y_6^0 &= A^2(K) \oplus Z_0 \oplus A(N) \oplus Z_{14}, \\
Y_6^1 &= A(A(N) \oplus Z_{14}), Y_6^2 = A(A(N) \oplus Z_{12}), Y_6^3 = A(Z_8), Y_6^4 = A(K), Y_6^5 = Z_9,
\end{aligned}$$

Let

$$Q = A(N). \tag{21}$$

Then, the expressions can be updated, as shown below:

$$\begin{aligned}
U_6^0 &= A^2(Q \oplus Z_2) \oplus A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10} \\
U_6^1 &= A(A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4) \oplus Q \oplus Z_2), \\
U_6^2 &= A(A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10})
\end{aligned}$$

$$\begin{aligned}
W_6^0 &= A^2(Q \oplus Z_6) \oplus Z_1 \oplus Q \oplus Z_{13}, \\
W_6^1 &= A(Q \oplus Z_{13}), \\
W_6^2 &= A(Q \oplus Z_{11}), W_6^3 = A(Q \oplus Z_5)
\end{aligned}$$

$$\begin{aligned}
Y_6^0 &= A^2(K) \oplus Z_0 \oplus Q \oplus Z_{14}, \\
Y_6^1 &= A(Q \oplus Z_{14}), Y_6^2 = A(Q \oplus Z_{12}), Y_6^3 = A(Z_8), Y_6^4 = A(K), Y_6^5 = Z_9,
\end{aligned}$$

When $r = 7$, we have

$$\begin{aligned}
U_7^0 &= A^2(A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10}) \oplus Z_0 \oplus A^2(Q \oplus Z_2) \oplus A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10} \\
&= A^2(A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10}) \oplus A^2(Q \oplus Z_2) \oplus A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4) \oplus Q \oplus Z_2 \\
U_7^1 &= A(A^2(Q \oplus Z_2) \oplus A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10}), \\
U_7^2 &= A(A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4) \oplus Q \oplus Z_2)
\end{aligned}$$

$$\begin{aligned}
W_7^0 &= A^2(Q \oplus Z_5) \oplus Z_1 \oplus A^2(Q \oplus Z_6) \oplus Z_1 \oplus Q \oplus Z_{13}, \\
W_7^1 &= A(A^2(Q \oplus Z_6) \oplus Z_1 \oplus Q \oplus Z_{13}), \\
W_7^2 &= A(Q \oplus Z_{13}), W_7^3 = A(Q \oplus Z_{11})
\end{aligned}$$

$$\begin{aligned}
Y_7^0 &= A(Z_9) \oplus Z_0 \oplus A^2(K) \oplus Z_0 \oplus Q \oplus Z_{14} \\
&= A(Z_9) \oplus A^2(K) \oplus Q \oplus Z_{14} \\
Y_7^1 &= A(A^2(K) \oplus Z_0 \oplus Q \oplus Z_{14}), \\
Y_7^2 &= A(Q \oplus Z_{14}), Y_7^3 = A(Q \oplus Z_{12}), Y_7^4 = A(Z_8), Y_7^5 = A(K),
\end{aligned}$$

5.1 Analyzing the Output of 8-Round Tiaoxin

As the current expressions are sufficiently complex, we will not write the expressions for the case $r = 8$. Since our aim is to analyze 8-round Tiaoxin, we need to focus on the output, as shown below:

$$\begin{aligned}
\mu_0 &= U_8^0 \oplus U_8^2 \oplus W_8^1 \oplus Y_8^3 \wedge W_8^3, \\
\mu_1 &= Y_8^0 \oplus W_8^2 \oplus U_8^1 \oplus Y_8^5 \wedge U_8^2.
\end{aligned}$$

In the following analysis, we assume that $Q[Diag(0)]$ traverses all the 2^{32} possible values and $Q[Diag(i)]$ $(i \neq 0)$ is assigned with a random constant.

As $U_8^1 = A(U_7^0)$ and N has passed through 4 AES rounds in the expression of U_7^0, we only focus on the integral property of μ_0. First, consider the quadratic part, as listed below:

$$Y_8^3 = Y_7^2 = A(Q \oplus Z_{14}),$$
$$W_8^3 = W_7^2 = A(Q \oplus Z_{13}).$$

Therefore, for the assumed input pattern of Q, $Y_8^3[Col(1,2,3)]$ and $W_8^3[Col(1,2,3)]$ are always constants. In other words, it can be derived that

$$\sum_{Q[Diag(0)] \in \mathbb{F}_2^{32}} (Y_8^3 \wedge W_8^3)[Col(1,2,3)] \quad = \quad 0.$$

As the algebraic degree of one AES round is 7, from the perspective of the algebraic degree, we indeed have

$$\sum_{Q[Diag(0)] \in \mathbb{F}_2^{32}} (Y_8^3 \wedge W_8^3) \quad = \quad 0. \tag{22}$$

Then, it is necessary to analyze $W_8^1 = A(W_7^0)$. As

$$W_7^0 \quad = \quad A^2(Q \oplus Z_5) \oplus Z_1 \oplus A^2(Q \oplus Z_6) \oplus Z_1 \oplus Q \oplus Z_{13},$$

according to Property 3, it can be known that the same value of

$$A^2(Q \oplus Z_5) \oplus Z_1 \oplus A^2(Q \oplus Z_6)$$

will appear an even number of times for the assumed input pattern of Q. Hence, it can be derived that the same value of $W_8^1[Col(1,2,3)]$ will appear an even number of times. In other words,

$$\sum_{Q[Diag(0)] \in \mathbb{F}_2^{32}} W_8^1[Col(1,2,3)] \quad = \quad 0. \tag{23}$$

Next, we are required to analyze $U_8^0 = A(U_7^2) \oplus U_7^0 \oplus Z_0$. For better understanding, the integral property of U_7^0 and $A(U_7^2)$ will be separately discussed. From the expression of U_7^0, it is easy to know that

$$\sum_{Q[Diag(0)] \in \mathbb{F}_2^{32}} A^2(Q \oplus Z_2) \oplus A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4) \oplus Q \oplus Z_2 \quad = \quad 0.$$

For the integral property of $A^2(A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10})$, we can first make $Q' = A(Q \oplus Z_4)$. Then, the set of Q' can be divided into 2^{24} different subsets according to the value of $(Q'[1][0], Q'[2][0], Q'[3][0])$, though the division depends on the key. In this way, for each subset of Q', it can be deduced that $A(A(Q') \oplus Q \oplus Z_{10})[i][j] = \mathcal{A}$ for $(0 \leq i \leq 3, 1 \leq j \leq 3)$. Therefore, it can be further derived that

$$\sum_{Q'[0][0] \in \mathbb{F}_2^{8}} \mathsf{SR} \circ \mathsf{SB} \circ A(A(Q') \oplus Q \oplus Z_{10})[ShiftCol(1,2,3)] \quad = \quad 0,$$

thus resulting in

$$\sum_{Q[Diag(0)] \in \mathbb{F}_2^{32}} \mathsf{SR} \circ \mathsf{SB} \circ A(A^2(Q \oplus Z_4) \oplus Q \oplus Z_{10})[ShiftCol(1,2,3)] \quad = \quad 0.$$

Consequently, we have

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(U_7^0)[ShiftCol(1,2,3)] \quad = \quad 0.$$

For U_7^2, similar to the analysis of W_8^1, it can be known that the same value of $U_7^2[Col(1,2,3)]$ will appear an even number of times. As a result, we have

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(A(U_7^2))[ShiftCol(1,2,3)] \quad = \quad 0.$$

Therefore, we can deduce that

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(U_8^0)[ShiftCol(1,2,3)] \quad = \quad 0. \tag{24}$$

Finally, it is essential to analyze the integral property of $U_8^2 = U_7^1$. However, we find that

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} A(A^2(Q\oplus Z_2)\oplus A^2(Q\oplus Z_3)\oplus A^2(Q\oplus Z_4)\oplus Q\oplus Z_{10})$$

is uncertain even though a very similar form will have an integral property which has been discussed for W_8^1. According to a similar conditional integral property as specified in Equation 5, if $Z_2[Diag(0)] = Z_3[Diag(0)]$ or $Z_3[Diag(0)] = Z_4[Diag(0)]$ or $Z_2[Diag(0)] = Z_4[Diag(0)]$ holds, assuming that it is $Z_2[Col(0)] = Z_3[Col(0)]$, then $A^2(Q\oplus Z_2)\oplus A^2(Q\oplus Z_3)$ is constant for the assumed input pattern of Q. Consequently, we only need to evaluate $A(A^2(Q\oplus Z_4)\oplus Q)$ and obviously there will be

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} A(A^2(Q\oplus Z_4)\oplus Q)[Col(1,2,3)] = 0.$$

Indeed, based on Property 5, there will be

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} A(A^2(Q\oplus Z_4)\oplus Q) = 0.$$

Formally, the condition $Z_2[Diag(0)] = Z_3[Diag(0)]$ corresponds to a set of weak keys satisfying

$$A^2(K)[Diag(0)] \quad = \quad Z_0[Diag(0)]. \tag{25}$$

The condition $Z_3[Diag(0)] = Z_4[Diag(0)]$ corresponds to a set of weak keys satisfying

$$A(K)[Diag(0)] \quad = \quad Z_0[Diag(0)]. \tag{26}$$

The condition $Z_2[Diag(0)] = Z_4[Diag(0)]$ corresponds to a set of weak keys satisfying

$$A^2(K)[Diag(0)] \quad = \quad A(K)[Diag(0)]. \tag{27}$$

When a weak key is used, there must be

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} U_8^2 = 0. \tag{28}$$

However, due to the influence of the integral property of U_8^0 as specified in Equation 24, we will lose the integral property for the output μ_0 without applying a linear transform

MC^{-1} to μ_0. As MC^{-1} is a linear transform, based on Equation 22, Equation 23 and Equation 28, it can be simply derived that

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(Y_8^3 \wedge W_8^3) = 0, \tag{29}$$

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(W_8^1)[Col(1,2,3)] = 0, \tag{30}$$

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(U_8^2) = 0. \tag{31}$$

Hence, when a key satisfies any of the three conditions specified in Equation 25, Equation 26 and Equation 27, combined with Equation 24, an integral property for μ_0 can be derived, as shown below:

$$\sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(\mu_0)[i][j] = \sum_{Q[Diag(0)]\in\mathbb{F}_2^{32}} \mathsf{MC}^{-1}(U_8^0 \oplus U_8^2 \oplus W_8^1 \oplus Y_8^3 \wedge W_8^3)[i][j] = 0,$$

where $(i,j) \in \{(0,1),(0,2),(0,3),(1,1),(1,2),(2,1),(2,3),(3,2),(3,3)\}$.

The distinguisher and weak keys. From the above dedicated manual analysis, there are 9 balanced bytes in $\mathsf{MC}^{-1}(\mu_0)$ when Q takes the assumed input pattern. As $Q = A(N)$, N can be computed from Q without the knowledge of the key K. Therefore, to make Q take the assumed input pattern, we simply assign values to Q such that they can form the assumed input pattern and then compute the corresponding set of N. In the distinguishing phase, we simply encrypt the set of N and observe the integral property of $\mathsf{MC}^{-1}(\mu_0)$. Therefore, the time complexity and data complexity to distinguish 8-round Tiaoxin are both 2^{32} when a weak key is used. For weak keys, there are 3 sets with 2^{96} keys each. Similarly, if imposing the condition on different diagonals, there will be in total $3 \times 4 = 12$ sets of weak keys with 2^{96} keys each.

Remark. It can be observed that what dominates the integral property of μ_0 will be the dedicated analysis of U_8^0 and U_8^2. From the evaluation of U_8^0, we learned that we need to evaluate $\mathsf{MC}^{-1}(\mu_0)$ rather than μ_0. From the evaluation of U_8^2, we learned that we need to add proper conditions in order to obtain an integral property. In addition, from our process to write the expressions, we found that we could arbitrarily choose an input pattern for $A(N)$ rather than N, which is equivalent to obtaining a free round and implies that one round is useless in Tiaoxin.

5.2 Feasibility of the Key-Recovery Attacks

As our key-recovery attack on 5-round AEGIS-128 succeeds, it is natural to ask whether it is also feasible to recover the weak key efficiently for 8-round Tiaoxin. One main reason why we can mount a key-recovery attack on 5-round AEGIS-128 is that an intermediate variable $T = A(N \oplus K)$ is introduced and the integral property of the output can be determined when only one byte of T is $\mathcal{A}$ and the remaining bytes are $\mathcal{C}$. As a result, we need to evaluate whether the same strategy can be applied to 8-round Tiaoxin.

First of all, we need to understand why the 8-round distinguisher requires 2^{32} data. One main reason arises in the evaluation of $U_8^0 = A(U_7^2) \oplus U_7^0 \oplus Z_0$ as Q will pass through 4 AES rounds both in $A(U_7^2)$ and U_7^0. However, different expressions in terms of Q appear in U_7^0 and U_7^2, which are $A^2(Q \oplus Z_3)$ and $A^2(Q \oplus Z_4)$. Although there are three choices of the conditions that the weak keys should satisfy, if choosing the condition that

$Z_3[Diag(0)] = Z_4[Diag(0)]$, the expressions of U_7^2 and U_7^1 can be significantly simplified if only $Q[Diag(0)]$ varies, as shown below:

$$\begin{aligned} U_7^1 &= A(A^2(Q \oplus Z_2) \oplus Z_{15} \oplus Q \oplus Z_{10}), \\ U_7^2 &= A(Z_{15} \oplus Q \oplus Z_2), \end{aligned}$$

where $Z_{15} = A^2(Q \oplus Z_3) \oplus A^2(Q \oplus Z_4)$ is a constant.

In this way, only in the expression of U_7^0, Q will pass through 4 AES rounds. Thus, it is necessary to introduce a new variable G to represent $A(Q \oplus Z_4)$ in order to mount a key-recovery attack, i.e.

$$G = A(Q \oplus Z_4).$$

In this way, the expression of U_7^0 can be updated as follows:

$$U_7^0 = A^2(A(G) \oplus Q \oplus Z_{10}) \oplus A^2(Q \oplus Z_2) \oplus Z_{15} \oplus Q \oplus Z_2.$$

Therefore, based on the relation between Q and G, it can be derived that

$$\sum_{G[0][0] \in \mathbb{F}_2^8} \mathsf{MC}^{-1}(U_7^0)[shiftCol(1,2,3)] = 0.$$

According to the expression of U_7^2, we have

$$\begin{aligned} & \sum_{G[0][0] \in \mathbb{F}_2^8} \mathsf{MC}^{-1}(A(U_7^2))[shiftCol(1,2,3)] \\ = & \sum_{G[0][0] \in \mathbb{F}_2^8} \mathsf{MC}^{-1}(A(A(Z_{15} \oplus Q \oplus Z_2)))[shiftCol(1,2,3)] = 0. \end{aligned}$$

As a result, we have

$$\sum_{G[0][0] \in \mathbb{F}_2^8} \mathsf{MC}^{-1}(U_8^0)[shiftCol(1,2,3)] = 0.$$

For the quadratic part $Y_8^3 \wedge W_8^3$, it remains that

$$\sum_{G[0][0] \in \mathbb{F}_2^8} (Y_8^3 \wedge W_8^3)[Col(1,2,3)] = 0.$$

Weak Constants. For $U_8^2 = U_7^1$, if Z_2 is treated as independent of Z_4, we immediately lose the integral property for U_8^2. As a result, we try to further reduce the size of weak keys by adding the condition $Z_4[i][i] = Z_2[i][i]$ for $1 \leq i \leq 3$. In this way, the key should satisfy the following conditions:

$$\begin{aligned} A(K)[Diag(0)] &= Z_0[Diag(0)], \\ A^2(K)[i][i] &= A(K)[i][i] \ (i \in \{1,2,3\}). \end{aligned}$$

It should be noted that when $Z_4[i][i] = Z_2[i][i]$ hold for $1 \leq i \leq 3$ and only $G[0][0] = A(Q \oplus Z_4)[0][0]$ is $\mathcal{A}$, it can be found that the set of $Q[Diag(0)]$ will be identical to the set of $(Q \oplus Z_2)[Diag(0)]$, thus resulting that $A(Q \oplus Z_2)[0][0]$ will be $\mathcal{A}$ and the remaining bytes will be $\mathcal{C}$. What we want to emphasize is that there is no need to add a stronger condition like $Z_4[Diag(0)] = Z_2[Diag(0)]$ to make $A(Q \oplus Z_2)[Col(0)] = G[Col(0)]$. One may also find that there are 4 possible ways to add conditions on Z_4 and Z_2 in order to make that only $A(Q \oplus Z_2)[0][0]$ is $\mathcal{A}$. However, if $Z_4[0][0] = Z_2[0][0]$ is involved, there is immediately a condition on the constant Z_0, i.e. $A(Z_0)[0][0] = Z_0[0][0]$. Therefore, we only focus on the way to choose conditions such that there are no additional conditions on the constant Z_0.

Let $K' = A(K)$ and we will have

$$\begin{aligned} K'[Diag(0)] &= Z_0[Diag(0)], \\ A(K')[i][i] &= K'[i][i] \ (i \in \{1,2,3\}), \end{aligned}$$

which implies that the number of weak keys becomes $2^{24\times 3} = 2^{72}$.

Once the key satisfies the above conditions, when only $G[0][0]$ is $\mathcal{A}$ and the remaining bytes of G are all $\mathcal{C}$, we can know that only $A(Q \oplus Z_2)[0][0]$ is $\mathcal{A}$ and the remaining bytes of $A(Q \oplus Z_2)$ are all $\mathcal{C}$. Therefore, according to the expression of U_7^1, we can know that

$$\sum_{G[0][0]\in\mathbb{F}_2^8} U_8^2[Col(1,2,3)] = 0.$$

For $W_8^1 = A(W_7^0)$, we also lose its integral property. However, if adding the condition $Z_5[Diag(0)] = Z_6[Diag(0)]$, when only $Q[Diag(0)]$ varies, $A^2(Q \oplus Z_5) \oplus A^2(Q \oplus Z_6)$ is a constant and we denote it by Z_{16}. In other words, when the following condition holds,

$$A(K)[Diag(0)] = Z_1[Diag(0)], \tag{32}$$

W_7^0 can be written as

$$W_7^0 = Z_1 \oplus Z_{16} \oplus Z_1 \oplus Q \oplus Z_{13}.$$

As a result,

$$\sum_{G[0][0]\in\mathbb{F}_2^8} W_8^1 = 0.$$

However, combined with the condition on the key, the condition specified in Equation 32 implies that

$$Z_0[Diag(0)] = Z_1[Diag(0)]. \tag{33}$$

In other words, if the constants Z_0 and Z_1 satisfy Equation 33 and the key satisfies

$$\begin{aligned} A(K)[Diag(0)] &= Z_0[Diag(0)], \\ A^2(K)[i][i] &= A(K)[i][i] \ (i \in \{1,2,3\}), \end{aligned}$$

there is an integral distinguisher for μ_0 as shown below:

$$\sum_{G[0][0]\in\mathbb{F}_2^8} \mathsf{MC}^{-1}(\mu_0)[i][j] = 0, \tag{34}$$

where $(i,j) \in \{(0,1),(0,2),(0,3),(1,1),(1,2),(2,1),(2,3),(3,2),(3,3)\}$. As $G = A(Q \oplus Z_4)$ and $Z_4 = Z_0 \oplus K$, $K[Diag(0)]$ can be simply recovered as in the key-recovery attack on 5-round AEGIS-128.

Recovering $K[Diag(0)]$. As $A(K)[Diag(0)] = Z_0[Diag(0)]$, we can first compute and store all the 2^{24} possible values for $K[Diag(0)]$ satisfying $A(K)[0][0] = Z_0[0][0]$. Denote the table storing $K[Diag(0)]$ by KT_1. Then, the procedure to recover the correct $A(K)[Diag(0)]$ is as follows:

Step 1: Construct a set of size 2^8 for G by traversing $G[0][0]$ while $G[i][j]$ $(i,j) \neq (0,0)$ is set as a random constant.

Step 2: Assign random values to $Q[Diag(i)]$ $(i = 1, 2, 3)$.

Step 3: For each candidate of $K[Diag(0)]$ in KT_1, construct 2^8 different values of N. Specifically, compute the 2^8 different values for $Q[Diag(0)]$ based on $G = A(Q \oplus K \oplus Z_0)$. Then, we compute the corresponding 2^8 different values of N based on $Q = A(N)$. Encrypt the 2^8 different values of N and collect the corresponding 2^8 outputs μ_0. If

$$\sum_{G[0][0]\in\mathbb{F}_2^8} \mathsf{MC}^{-1}(\mu_0)[i][j] \quad = \quad 0,$$

where $(i,j) \in \{(0,1),(0,2),(0,3),(1,1),(1,2),(2,1),(2,3),(3,2),(3,3)\}$, then the current candidate for $K[Diag(0)]$ is the correct value and exit. Otherwise, consider the next candidate for $K[Diag(0)]$ and repeat.

Complexity evaluation. As there are 2^{24} candidates for $K[Diag(0)]$, the data and time complexity are both upper bounded by $2^{24+8} = 2^{32}$. To store the candidates of $K[Diag(0)]$, the memory complexity is 2^{24}.

Experiments. The experiments are performed on the small-scale AES [CMR05]. For both the distinguishing attack and the key-recovery attack using a weak constant, experiments show that they succeed with probability 1. If any condition on the key does not hold, both the attacks fail. If a weak constant is not used, the key-recovery attack also fails. In addition, we observe that the whole W_8^1 rather than only $W_8^1[Col(1,2,3)]$ are balanced in the distinguishing attack. However, it should be emphasized that this will not happen when the AES round function is used. A detailed discussion can be found at Appendix A.

Efficiently recovering the full key. After $K[Diag(0)]$ is known, $A(K)[Col(0)]$ is known. As $A(K)[Diag(0)] = Z_0[Diag(0)]$, we indeed can know both $A(K)[Col(0)]$ and $A(K)[Diag(0)]$. To compute $A(K)[Diag(i)]$ $(1 \le i \le 3)$, we can simply exhaust all the 2^{24} possible values of $A(K)[Diag(i)]$ and compute $A^2(K)[Col(i)]$ and check whether $A^2(K)[i][i] = A(K)[i][i]$ holds. Hence, there will be 2^{16} possible candidates for $A(K)[Diag(i)]$ $(1 \le i \le 3)$. Consequently, there are in total 2^{48} possible values of $A(K)$. In other words, the time complexity to recover the weak key is 2^{48}, which is obviously much smaller than the number of weak keys, i.e. 2^{72}.

Feasibility for 8-round Tiaoxin. As the above analysis shows, to mount a key-recovery attack on 8-round Tiaoxin, the size of the set of the weak keys will be reduced to 2^{72} and there are 32 bit conditions on the constants (Z_0, Z_1). Obviously, the constants used in Tiaoxin does not satisfy Equation 33 and thus the above key-recovery attack cannot be applied to 8-round Tiaoxin. However, it is an alarm that in the design like Tiaoxin, the round constants should be carefully chosen.

6 Discussions

The basic idea to attack 5-round AEGIS-128 and 8-round Tiaoxin is simple, which is to utilize the conventional integral distinguisher for 4-round AES. However, the feasibility of the simple idea needs to be highlighted, which can obviously advance the understanding of AEGIS-128 and Tiaoxin.

Feasibility for 5-round AEGIS-128. For 5-round AEGIS-128, the feasibility much relies on the fact that X_5^0 is not involved in the output, where N will pass through 5 AES rounds. For the distinguishing attack, it is essential to consider the weak key. However, the weak key is not obvious and hidden in the expressions of the internal states. Therefore, our way to write and analyze the expressions plays an important role to identify the weak keys. For the key-recovery attack, the feasibility also contributes to the analysis of the expressions as we find that it is possible to introduce a variable T to replace $A(K \oplus N)$, which directly makes the variable N disappear in almost all expressions. In other words, introducing T is equivalent to appending a round for key recovery. While appending several rounds before a distinguisher for key recovery is common in the analysis of block ciphers, it is obviously non-intuitive for AEGIS-128.

Feasibility for 8-round Tiaoxin. It is common in the cryptanalysis of symmetric-key primitives to add conditions to reduce the algebraic degree. Therefore, the condition on the keys for 5-round AEGIS-128 is still traceable if more attention is paid. However, the condition on the key for 8-round Tiaoxin appears for a very different reason, which we believe hard to detect without writing and analyzing the expressions of the internal states. Especially, from our process to write the expressions until the 6th round, we find that the expressions can be represented in terms of $A(N)$ rather than N. By replacing $A(N)$ with a new variable Q, instead of considering N itself, we can directly study Q as computing N from Q requires no secret knowledge, which implies that there is 1 useless round in Tiaoxin. To mount a key-recovery attack, we have to add more conditions as there are several different terms like $A(Q \oplus \nu_i)$ where ν_i represents a different 128-bit constant for different i. However, by carefully studying the expressions, we find it still feasible to mount a key-recovery attack when a weak constant is used, which occurs with probability 2^{-32}. In summary, lots of useful information related to attacks is hidden in the expressions and it is necessary to perform dedicated analysis of them.

6.1 On the Usage of Division Property

The bit-based division property (BDP) [TM16] is a popular tool to search for integral distinguishers, especially when equipped with the automatic tools [XZBL16]. However, without the identification of the weak keys with our methods, a naive implementation without taking the conditions on the key into account will obviously fail in finding the integral distinguishers for 5-round AEGIS-128 and 8-round Tiaoxin. The reason is that these distinguishers will not hold when a non-weak key is used. Especially for Tiaoxin, the input pattern of N is unstructured as we indeed consider a structured pattern of $A(N)$. Without noticing this fact, it is impossible to find the distinguisher with 2^{32} data complexity with the naive implementation of BDP.

Only for interest, we also tested whether the conventional bit-based division property (CBDP) [TM16] can detect the phenomenon when the same value appears an even number of times in a multiset. When saying CBDP, we mean that the integral property is evaluated based on whether there exists a feasible desired division trail [XZBL16].

We implemented CBDP to evaluate the integral property for

$$\sum_{s[Diag(0)] \in \mathbb{F}_2^{32}} R(R^2(s \oplus c_0) \oplus R^2(s \oplus c_1)).$$

It has been proved in Property 3 that the sum must be zero. However, it is evaluated as unknown with CBDP. It should be mentioned that the sum

$$\sum_{s[Diag(0)] \in \mathbb{F}_2^{32}} R^2(s \oplus c_0) \oplus R^2(s \oplus c_1)$$

is correctly predicted, which is obviously reasonable as

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}} R^2(s\oplus c_0)=0,\qquad \sum_{s[Diag(0)]\in\mathbb{F}_2^{32}} R^2(s\oplus c_1)=0.$$

To explain why CBDP failed, we will construct a special example. Specifically, consider a mapping $\varphi(v_0,v_1,v_2,\kappa_0,\kappa_1,\kappa_2,\kappa_3,\kappa_4,\kappa_5):\mathbb{F}_2^9\to\mathbb{F}_2^3$ and denote the three output bits by $(\iota_0,\iota_1,\iota_2)$, as defined below:

$$\begin{aligned}\iota_0 &= (v_0\oplus\kappa_0)(v_1\oplus\kappa_2)\oplus(v_0\oplus\kappa_1)(v_1\oplus\kappa_3),\\ \iota_1 &= (v_0\oplus\kappa_0)(v_2\oplus\kappa_4)\oplus(v_0\oplus\kappa_1)(v_2\oplus\kappa_5),\\ \iota_2 &= (v_1\oplus\kappa_2)(v_2\oplus\kappa_4)\oplus(v_1\oplus\kappa_3)(v_2\oplus\kappa_5).\end{aligned}$$

Obviously, for an arbitrary choice of $(\kappa_0,\kappa_1,\kappa_2,\kappa_3,\kappa_4,\kappa_5)$, there must be

$$\sum_{(v_0,v_1,v_2)\in\mathbb{F}_2^3}\iota_0=0,\qquad \sum_{(v_0,v_1,v_2)\in\mathbb{F}_2^3}\iota_1=0,\qquad \sum_{(v_0,v_1,v_2)\in\mathbb{F}_2^3}\iota_2=0.$$

It is certain that CBDP can predict these integral properties.

Moreover, from the above construction, it can be observed that

$$\begin{aligned}&\varphi(v_0,v_1,v_2,\kappa_0,\kappa_1,\kappa_2,\kappa_3,\kappa_4,\kappa_5)\\ =\ &\varphi(v_0\oplus\kappa_0\oplus\kappa_1,v_1\oplus\kappa_2\oplus\kappa_3,v_2\oplus\kappa_4\oplus\kappa_5,\kappa_0,\kappa_1,\kappa_2,\kappa_3,\kappa_4,\kappa_5).\end{aligned}$$

Therefore, when (v_0,v_1,v_2) traverses all the possible 8 values and $(\kappa_0,\kappa_1,\kappa_2,\kappa_3,\kappa_4,\kappa_5)$ are set as constants, in the obtained multiset of $(\iota_0,\iota_1,\iota_2)$, the same value of $(\iota_0,\iota_1,\iota_2)$ must appear an even number of times. As a result, for an arbitrary boolean function $g(\iota_0,\iota_1,\iota_2):\mathbb{F}_2^3\to\mathbb{F}_2$, there must be

$$\sum_{(v_0,v_1,v_2)\in\mathbb{F}_2^3} g(\iota_0,\iota_1,\iota_2)=0.$$

Then, the question becomes whether CBDP can capture it. To show that CBDP cannot capture it, it is sufficient to find a feasible division trail ending with "1". Therefore, we consider a quadratic boolean function $g(\iota_0,\iota_1,\iota_2)=\iota_2\wedge(\iota_0\oplus\iota_1)$. The circuit to calculate g is depicted in Figure 6. Based on the propagation rules, we can deduce by hand a feasible division trail ending with "1" as illustrated in Figure 6, thus revealing that CBDP cannot predict the sum of g. Consequently, it is an evidence that CBDP is unable to capture the property that the same value appears an even number of times in a multiset. Although the polynomials are known in this special example and can indeed be simplified, i.e. the circuit will change after simplification, it is too complex to write the boolean expression of an output bit for a cryptographic primitive and therefore the evaluation is indeed based on the (non-simplified) circuit to compute the output. This is why we directly study the circuit rather than the accurate simplified expression of $\iota_2\wedge(\iota_0\oplus\iota_1)$ in this special example.

A potential method to address this problem is to use the recent new ideas [WHG+19, HLM+20, HSWW20], which is to count the number of division trails. When it is even, the sum is treated as zero, which fits very well with our theoretical analysis as we prove that the same value must appear an even number of times in a multiset. However, due to the influence of the matrix multiplication of AES, as revealed in [HLLT20], the number of trails will explode when the number of matrix multiplications increases. Thus, it is questionable whether the solver can enumerate all the feasible solutions in practical time, especially when evaluating

$$\sum_{s[Diag(0)]\in\mathbb{F}_2^{32}} R^n(R^2(s\oplus c_0)\oplus R^2(s\oplus c_1))$$

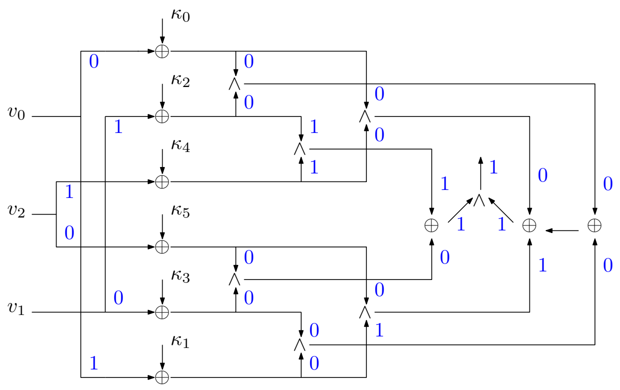

Figure 6: The tested example

for large n, which implies the importance to prove Property 3.

Combining all the above discussions, it seems plausible why the distinguishers for 5-round AEGIS-128 and 8-round Tiaoxin with data complexity 2^{32} in the weak-key setting are not found during the long CAESAR competition. Moreover, the feasibility to append additional rounds for key recovery is deeply hidden in the expressions of the internal states.

7 Conclusion

By expressing the internal states in terms of the input state words, we observed the possibility to adapt the well-known integral distinguisher on reduced AES to AEGIS-128 and Tiaoxin in the weak-key setting. With dedicated analysis of these expressions, the set of weak keys are eventually identified, which are used to simplify the quadratic part of the output for AEGIS-128 and to turn a probabilistic integral property into a deterministic one for Tiaoxin, respectively. To make the derived integral distinguisher theoretically correct, we have proved some integral properties for some unusual combinations of the AES round function, which will easily occur in the constructions like AEGIS and Tiaoxin but will never occur in real AES. To efficiently recover the weak key, we introduce a new variable related to the key to represent the output of the AES round function for a certain 128-bit word and then study the updated expressions. Such a way is almost equivalent to appending rounds for key-recovery before a distinguisher and the feasibility much relies on the careful analysis of the updated expressions. Consequently, distinguishing and key-recovery attacks on 5-round AEGIS-128 are achieved in the weak key setting. For 8-round Tiaoxin, we could only construct the distinguisher, while the key-recovery attack requires the usage of a weak constant occurring with probability 2^{-32}. This is the first third-party cryptanalysis of the initialization phase for both AEGIS-128 and Tiaoxin and all the attacks reach half of the total number of rounds. Based on our analysis, it seems that attacks on constructions like AEGIS-128 and Tiaoxin in the weak-key setting have more potential.

Acknowledgements. We thank the anonymous reviewers of ToSC 2021 Issue 2 for their useful comments. Especially, we would like to thank a reviewer for sharing another proof of Property 1, which inspired us to simplify the proof. Fukang Liu is supported by Invitation Programs for Foreigner-based Researchers of NICT. In addition, he is also partially supported by the National Natural Science Foundation of China (No. 62072181) and the International Science and Technology Cooperation Projects (No. 61961146004). Takanori Isobe is supported by JST, PRESTO Grant Number JPMJPR2031, Grant-in-Aid for Scientific Research (B)(KAKENHI 19H02141), and Support Center for Advanced

TelecommunicationsTechnology Research (SCAT). Kosei Sakamoto is supported by Grant-in-Aid for JSPS Fellows (KAKENHI 20J23526) for Japan Society for the Promotion of Science.

References

[AEL$^+$18] Tomer Ashur, Maria Eichlseder, Martin M. Lauridsen, Gaëtan Leurent, Brice Minaud, Yann Rotella, Yu Sasaki, and Benoît Viguier. Cryptanalysis of MORUS. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 35–64. Springer, 2018.

[BB93] Ishai Ben-Aroya and Eli Biham. Differential cryptanalysis of lucifer. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 1993.

[Bey18] Tim Beyne. Block cipher invariants as eigenvectors of correlation matrices. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 3–31. Springer, 2018.

[BS01] Alex Biryukov and Adi Shamir. Structural cryptanalysis of SASAS. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2001.

[CMR05] Carlos Cid, Sean Murphy, and Matthew J. B. Robshaw. Small scale variants of the AES. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 145–162. Springer, 2005.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Information Security and Cryptography. Springer, 2002.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.

[ENP19] Maria Eichlseder, Marcel Nageler, and Robert Primas. Analyzing the linear keystream biases in AEGIS. *IACR Trans. Symmetric Cryptol.*, 2019(4):348–368, 2019.

[GJN+16] Jian Guo, Jérémy Jean, Ivica Nikolic, Kexin Qiao, Yu Sasaki, and Siang Meng Sim. Invariant subspace attack against midori64 and the resistance criteria for s-box designs. *IACR Trans. Symmetric Cryptol.*, 2016(1):33–56, 2016.

[HLLT20] Phil Hebborn, Baptiste Lambin, Gregor Leander, and Yosuke Todo. Lower bounds on the degree of block ciphers. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 537–566. Springer, 2020.

[HLM+20] Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 466–495. Springer, 2020.

[HSWW20] Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 446–476. Springer, 2020.

[HWX+17] Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. Conditional cube attack on reduced-round keccak sponge function. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 259–288, 2017.

[JN16] Jérémy Jean and Ivica Nikolic. Efficient design strategies based on the AES round function. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 334–353. Springer, 2016.

[KMN10] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of nlfsr-based cryptosystems. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010.

[KW02] Lars R. Knudsen and David A. Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.

[LAAZ11] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A cryptanalysis of printcipher: The invariant subspace attack. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2011.

[LMR15] Gregor Leander, Brice Minaud, and Sondre Rønjom. A generic approach to invariant subspace attacks: Cryptanalysis of robin, iscream and zorro. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 254–283. Springer, 2015.

[Min14] Brice Minaud. Linear biases in AEGIS keystream. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2014.

[Nik] Ivica Nikolić. Tiaoxin. http://competitions.cr.yp.to/round3/tiaoxinv21.pdf.

[SSS+19] Danping Shi, Siwei Sun, Yu Sasaki, Chaoyun Li, and Lei Hu. Correlation of quadratic boolean functions: Cryptanalysis of all versions of full \mathsf MORUS. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 180–209. Springer, 2019.

[TLS16] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear invariant attack - practical attack on full scream, iscream, and midori64. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 3–33, 2016.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 357–377. Springer, 2016.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015.

[VV18] Serge Vaudenay and Damian Vizár. Can caesar beat galois? - robustness of CAESAR candidates against nonce reusing and high data complexity attacks. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography* *and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, volume 10892 of *Lecture Notes in Computer Science*, pages 476–494. Springer, 2018.

[WHG+19] Senpeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. Milp-aided method of searching division property using three subsets and applications. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 398–427. Springer, 2019.

[WLF+05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.

[WP13] Hongjun Wu and Bart Preneel. AEGIS: A fast authenticated encryption algorithm. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 185–201. Springer, 2013.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.

[ZXL17] Wenying Zhang, Zhen Xiao, and Mengzhu Li. Optimised the differential trail of the authenticated encryption algorithm tiaoxin-346. *Int. J. High Perform. Comput. Netw.*, 10(6):498–504, 2017.

A Experimental Findings

In our experiments for the distinguisher on 8-round Tiaoxin, we observe that the whole W_8^1 rather than only $W_8^1[Col(1,2,3)]$ are balanced. As the experiments are performed with the small-scale AES round function, it is necessary to identify whether this will happen when the real AES round function is used. Note that

$$W_8^1 \quad = \quad A(A^2(Q \oplus Z_5) \oplus A^2(Q \oplus Z_6) \oplus Q \oplus Z_{13}),$$

where Z_5, Z_6 and Z_{13} are key-dependent constants.

Hence, we are motivated to investigate whether

$$\sum_{s[Diag(i)] \in \mathbb{F}_2^{32}} A(A^2(s) \oplus s \oplus c_0 \oplus A^2(s \oplus c_1)) = 0 \tag{35}$$

always holds for two arbitrary 128-bit constants (c_0, c_1).

Although Equation 35 is very similar to Equation 4 and Equation 7, we are unable to find a similar way to prove it. Although it is trivial to deduce from Property 3 that

$$\sum_{s[Diag(0)] \in \mathbb{F}_2^{32}} R(R^2(s) \oplus s \oplus c_0 \oplus R^2(s \oplus c_1))[Col(1,2,3)] = 0,$$

whether Equation 36 holds is unknown.

$$\sum_{s[Diag(0)] \in \mathbb{F}_2^{32}} R(R^2(s) \oplus s \oplus c_0 \oplus R^2(s \oplus c_1))[Col(0)] = 0. \tag{36}$$

Let $\tau = R^2(s) \oplus R^2(s \oplus c_1) \oplus s \oplus c_0$. From the proof of Property 3, it can be found that the expression of $\tau[0][0]$ can be written as follows:

$$\begin{aligned} \tau[0][0] \quad &= \quad 2 \cdot \mathsf{S}(2 \cdot \mathsf{S}(s[0][0]) \oplus 3 \cdot \mathsf{S}(s[1][1]) \oplus \mathsf{S}([2][2]) \oplus \mathsf{S}(s[3][3])) \\ &\oplus \quad 2 \cdot \mathsf{S}(2 \cdot \mathsf{S}(s[0][0] \oplus c_1[0][0]) \oplus 3 \cdot \mathsf{S}(s[1][1] \oplus c_1[1][1]) \\ &\oplus \quad \mathsf{S}([2][2] \oplus c_1[2][2]) \oplus \mathsf{S}(s[3][3] \oplus c_1[3][3])) \\ &\oplus \quad s[0][0] \oplus \varpi, \end{aligned}$$

where ϖ is a constant depending on c_0, c_1 and the constant part of s.

For the small-scale AES round function, we performed an exhaustive search. Specifically, for each value of $c_1[Diag(0)]$, which is 2^{16} in total for the small-scale AES, we traversed all the 2^{16} possible values for $s[Diag(0)]$ and collect the corresponding set of $\tau[0][0]$. It is found that the same value of $\tau[0][0]$ in the set always appears an even number of times. Then, the expressions for $\tau[i][i]$ $(1 \leq i \leq 3)$ can also be written. After performing a similar exhaustive search for each $\tau[i][i]$, it is also observed that the same value of it in the computed set always appears an even number of times, thus explaining why the whole W_8^1 is balanced in our experiments.

However, when we change it to the real AES round function, the exhaustive search is obviously infeasible. However, to disprove something, it suffices to find a counter-example. In the experiments, we randomly chose a value for $c_1[Diag(0)]$ and ϖ and evaluated the sum of the set of $\mathsf{S}(\tau[0][0])$ when $s[Diag(0)]$ traverses all the 2^{32} possible values. It is found the sum is always non-zero, thus disproving Equation 36. Similar experiments were also performed to independently evaluate the sum of $\mathsf{S}(\tau[i][i])$ $(1 \leq i \leq 3)$.