Iterative Differential Characteristic of TRIFLE-BC

Fukang Liu[1,3], Takanori Isobe[2,3]

[1] Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
liufukangs@163.com
[2] National Institute of Information and Communications Technology, Japan
[3] University of Hyogo, Hyogo, Japan
takanori.isobe@ai.u-hyogo.ac.jp

Abstract. TRIFLE is a Round 1 candidate of the NIST Lightweight Cryptography Standardization process. In this paper, we present an interesting 1-round iterative differential characteristic of the underlying block cipher TRIFLE-BC used in TRIFLE, which holds with probability of 2^{-3}. Consequently, it allows to mount distinguishing attack on TRIFLE-BC for up to 43 (out of 50) rounds with data complexity 2^{124} and time complexity 2^{124}. Most importantly, with such an iterative differential characteristic, the forgery attack on TRIFLE can reach up to 21 (out of 50) rounds with data complexity 2^{63} and time complexity 2^{63}. Finally, to achieve key recovery attack on reduced TRIFLE, we construct a differential characteristic covering three blocks by carefully choosing the positions of the iterative differential characteristic. As a result, we can mount key-recovery attack on TRIFLE for up to 11 rounds with data complexity 2^{63} and time complexity 2^{104}. Although the result in this paper cannot threaten the security margin of TRIFLE, we hope it can help further understand the security of TRIFLE.

Keywords: AEAD, TRIFLE, differential attack, distinguisher, forgery

1 Introduction

With the development of the emerging areas like sensor networks, healthcare, distributed control systems, the Internet of Things, and cyber physical systems, where highly-constrained devices are interconnected, typically communicating wirelessly with one another, and working in concert to accomplish some task, new requirements for cryptographic algorithms start to appear. The new requirements covers such aspects as energy, power, area and throughput. Consequently, recent years have witnessed many new designs of lightweight block ciphers, hash functions and stream ciphers like PRESENT [7], KATAN [8], PICCOLO [20], PHOTON [12], SIMON/SPECK [4], Midori [2], SKINNY [5], Plantlet [16], and QARMA [1], just to name a few. The main reason why there is a demand for lightweight designs lies in that conventional cryptographic algorithms designed for desktop/server environments cannot fit into constrained device. To standardize lightweight cryptographic algorithms that are suitable for use in constrained environments, the National Institute of Standards and Technology (NIST) started a public lightweight cryptography competition project in as early as 2013 and initiated the call for submissions in 2018, with the hope to select a lightweight cryptographic standard by combining the efforts of both academia and industry. The 56 Round 1 candidates of the NIST Lightweight Cryptography Standardization project became public on April 18, 2019. Since the publication, the cryptanalysis has started. For instance, a probability 1 iterative differential characteristic in the SNEIK permutation was identified in [18], which was quickly exploited to mount forgery attack on full SNEIKEN [13]. As a response for this attack, the designer of SNEIK has updated SNEIK accordingly. Very recently, Eichlseder et al. also showed a forgery attack on FlexAEAD [11].

Since there exist several advanced cryptanalysis techniques to evaluate the security of a primitive, the resistance against differential attack [6], linear attack [15], integral attack [14,22,23], cube attack [10] and many other attack methods have been well analyzed by the designers for most submitted primitives. The feasibility of efficient security evaluation against classical cryptanalysis is owing to the emerging automatical cryptanalysis techniques to model the corresponding attacks [17,19,21,24,25] and solve with existing state-of-the-art solvers.

TRIFLE is one of the Round 1 candidates [9]. It is an Authenticated Encryption with Associated Data (AEAD) algorithm, which is constructed based on the underlying block cipher TRIFLE-BC. TRIFLE employs a MAC-then-Encrypt type paradigm, where CBC style authentication is done on the nonce, associated data and the plaintext to generate the tag. This tag is then used as a random IV in an output feedback mode of encryption to generate the ciphertext. The authors make a security claim for TRIFLE under the IND-CPA and INT-CTXT security model, which requires that the data complexity cannot exceed 2^{64} and time complexity cannot exceed 2^{128}. In other words, the total number of blocks (among all messages and associated data) processed through the underlying block cipher for a fixed master key at the online phase cannot exceed 2^{64}. Moreover, the designers claim that the linear transform in TRIFLE-BC can provide the maximal diffusion. However, as will be shown, an iterative differential characteristic with hamming weight 1 is identified for TRIFLE-BC. Although such an iterative differential characteristic cannot be exploited to mount distinguishing attack on full TRIFLE-BC, we can attack 43 out of 50 rounds of TRIFLE-BC, revealing the powerful effect of such an iterative differential characteristic. In a word, when taking into account the combination of the S-box and linear transform in TRIFLE-BC, an interesting and surprising iterative differential characteristic with hamming weight 1 is identified, which holds with probability 2^{-3}. Some related analytical results are presented in Table 1.

Table 1: The analytical results of TRIFLE

Attack Type	Target	Rounds	Data	Time	Ref.
Distinguishing attack	TRIFLE-BC	43	2^{124}	2^{124}	Section 4.1
Forgery attack	TRIFLE	21	2^{63}	2^{63}	Section 4.2
Key-recovery attack	TRIFLE	11	2^{63}	2^{104}	Section 4.3
Key-recovery attack	TRIFLE-BC	44	2^{126}	2^{126}	Section 4.3

Organization This paper is organized as follows. The notations and the description of TRIFLE are given in Section 2. Then, we reveal the iterative differential characteristic of TRIFLE-BC in section 3. The application of this iterative differential characteristic on distinguishing attack, forgery attack and key-recovery attack is detailed in Section 4. Finally, we conclude the paper in Section 5.

2 Preliminaries

In this section, we give a description of the notations and the primitive TRIFLE.

2.1 Notation

For a better understanding of this paper, some notations are given below.

1. W represents the internal state of TRIFLE-BC.
2. W^i represents the input state of the i-th round of RIFLE-BC where $(0 \leq i \leq 49)$.
3. W_s^i/W_p^i respectively represent the internal state after **SubNibbles/BitPermutaion** in the i-th round of TRIFLE-BC.
4. $\varDelta W^i/\varDelta W_s^i/\varDelta W_p^i$ respectively represent the xor difference of W^i, W_s^i and W_p^i.
5. $Z[i]$ represents the i-th bit of Z (Z can be W^i, W_s^i, $W_p^i \cdots$). $Z[0]$ denotes the least significant bit of Z.
6. $Z[j \sim i]$ represents the j-th bit to the i-th bit of Z (Z can be W^i, W_s^i, $W_p^i \cdots$). For example, $Z[1 \sim 0]$ denotes the two bits $Z[1]$ and $Z[0]$.
7. $\ggg$ and $\oplus$ respectively represent the logic operation: rotate right and exclusive or.
8. $A||B$ represents the concatenation of A and B.

2.2 Specification of TRIFLE

TRIFLE is a block cipher based authenticated encryption mode with block size $n = 128$ that receives an encryption key $K \in \{0,1\}^{128}$, a nonce $N \in \{0,1\}^{128}$, an associated data $A \in \{0,1\}^*$ and a message $M \in \{0,1\}^*$ as inputs and returns a ciphertext $C \in \{0,1\}^{|M|}$ and a tag $T \in \{0,1\}^{128}$, where $|M|$ represents the length of M in number of bits. The underlying block cipher is denoted by TRIFLE-BC. TRIFLE employs a MAC-then-Encrypt type paradigm, where CBC style authentication is done on the nonce, associated data and the plaintext to generate the tag. This tag is then used as a random IV in an output feedback mode of encryption to generate the ciphertext. The construction of TRIFLE is depicted in Fig. 1, where **Prefix** is a constant to represent whether the associated data A and message M are empty. Besides, OZP is the function that applies optional 10^* padding on n bits, i.e., $OZP(X) = 0^{n-|X|-1}||1||X$ when $|X| < n$, and $OZP(X) = X$, if $|X| = n$. More details of TRIFILE can be found at [9].

The underlying block cipher TRIFLE-BC (E_K in Fig. 1) used in TRIFLE receives a 128-bit plaintext P and a 128-bit key $K = (K_7||K_6||...||K_0)$, where $K_i \in F_2^{16}$ $(0 \leq i \leq 7)$. The internal state of TRIFLE-BC W can be viewed as 32 4-bit nibbles. The block cipher TRIFLE-BC is composed of 50 rounds and each round consists of four consecutive operations **SubNibbles**, **BitPermutation**, **AddRoundKey** and **AddRoundConst**, as specified in the following. More details can be found at [9].

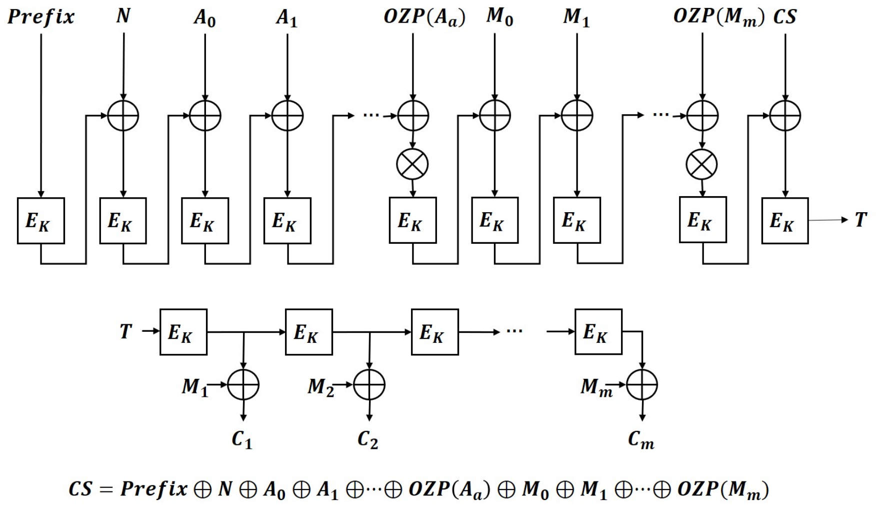

Prefix
N
A_0
A_1
$OZP(A_a)$
M_0
M_1
$OZP(M_m)$
CS
E_K
T
M_1
M_2
M_m
C_1
C_2
C_m
$CS = Prefix \oplus N \oplus A_0 \oplus A_1 \oplus \cdots \oplus OZP(A_a) \oplus M_0 \oplus M_1 \oplus \cdots \oplus OZP(M_m)$

Fig. 1: The construction of TRIFLE

SubNibbles TRIFLE-BC uses an invertible 4-bit S-box and applies it to every nibble of the internal state. The specification of this S-box can be found at Table 4. The corresponding Differential Distribution Table (DDT) can be found at Table 2.

BitPermutation TRIFLE-BC uses an optimal bit permutation to create maximal diffusion. This permutation maps bits from bit position i of the cipher state to bit position $P(i)$, where $P(i) = i/4 + (i \bmod 4) \times 32$. The bit mapping table can be seen at Table 3.

AddRoundKey The round key is xored with the interal state $(W[4i+1], W[4i+2])$ $(0 \leq i \leq 31)$. Specifically, suppose the 128-bit $K^i = (K_7^i \| K_6^i \| ... \| K_0^i)$ $(0 \leq i \leq 49)$ represents the key used in the i–th round. Then, at round i, the key addition is proceeded as follows.

$$U^i[31]\|...\|U^i[0] \leftarrow K_4^i \| K_5^i,$$
$$V^i[31]\|...\|V^i[0] \leftarrow K_1^i \| K_0^i,$$
$$W[4j+2] \leftarrow W[4j+2] \oplus U^i[j] \ (0 \leq j \leq 31),$$
$$W[4j+1] \leftarrow W[4j+1] \oplus V^i[j] \ (0 \leq j \leq 31).$$

As can be seen, we can also denote the i-th round key by (U^i, V^i), where $U^i \in F_2^{32}$ and $V^i \in F_2^{32}$. After key addition, the round key used for next round is generated as follows, which is similar to the one used in GIFT-128 [3].

$$K_7^{i+1}\|...\|K_0^{i+1} \leftarrow K_1^i \ggg 2 \| K_0^i \ggg 12 \| K_7^i \| ... \| K_2^i.$$

Table 2: DDT of S-box of TRIFLE-BC

Δ_{in}/Δ_{out}	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	2	2	2	0	2	0	4	2	2	0
2	0	2	0	0	0	2	0	0	0	4	0	2	2	2	2	0
3	0	2	2	0	2	2	0	4	0	2	0	2	0	0	0	0
4	0	0	2	4	0	0	0	2	0	2	2	2	0	2	0	0
5	0	0	0	0	0	4	2	2	0	2	2	0	0	2	0	2
6	0	0	2	2	2	0	0	2	2	0	2	0	0	0	4	0
7	0	0	2	2	4	0	0	0	0	2	0	2	2	0	0	2
8	0	0	0	2	2	2	4	2	0	0	0	2	0	0	2	0
9	0	2	2	0	0	0	0	0	2	0	2	4	2	2	0	0
a	0	0	0	2	0	2	0	0	0	0	4	2	2	0	2	2
b	0	2	4	0	0	0	2	0	0	2	0	0	2	2	0	2
c	0	2	0	0	2	2	2	0	2	0	0	0	0	4	2	0
d	0	4	0	2	0	0	2	0	2	0	0	0	2	0	2	2
e	0	0	0	2	2	0	2	2	4	2	0	0	0	0	0	2
f	0	2	2	0	2	2	0	0	2	0	2	0	0	0	0	4

Table 3: The TRIFLE-BC BitPermutation

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	32	64	96	1	33	65	97	2	34	66	98	3	35	67	99
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	36	68	100	5	37	69	101	6	38	70	102	7	39	71	103
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	40	72	104	9	41	73	105	10	42	74	106	11	43	75	107
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	44	76	108	13	45	77	109	14	46	78	110	15	47	79	111
i	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
$P(i)$	16	48	80	112	17	49	81	113	18	50	82	114	19	51	83	115
i	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
$P(i)$	20	52	84	116	21	53	85	117	22	54	86	118	23	55	87	119
i	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
$P(i)$	24	56	88	120	25	57	89	121	26	58	90	122	27	59	91	123
i	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
$P(i)$	28	60	92	124	29	61	93	125	30	62	94	126	31	63	95	127

AddRoundConst The 6-bit round constant is xored with the following 6 internal state bits: $W[23]$, $W[19]$, $W[15]$, $W[11]$, $W[7]$ and $W[3]$. Besides, a constant bit 1 is added to in the most significant bit $W[127]$. The 6-bit round constant is generated with the same 6-bit affine LFSR used in SKINNY [5].

Table 4: The S-box of TRIFLE-BC

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

2.3 Security Claim of TRIFLE

The designers of TRIFLE make a security claim for TRIFLE under IND-CPA and INT-CTXT security mode, as shown in Table 5. The data complexity of the attack quantifies the online resource requirements, and includes the total number of blocks (among all messages and associated data) processed through the underlying block cipher for a fixed master key. Therefore, the data complexity of our forgery attack and key-recovery attack are both below 2^{64}. Note that the security claims are valid in nonce-misuse scenario as well.

Table 5: Security claim of TRIFLE by the designers

Security Mode	Data Complexity	Time Complexity
IND-CPA	2^{64}	2^{128}
INT-CTXT	2^{64}	2^{128}

3 1-Round Iterative Differential Characteristic of TRIFLE-BC

The 1-round iterative differential characteristic of TRIFLE-BC is found with an MILP method based on Sun et al's work [21]. Actually, even without such an MILP-based method, such a result can be obtained as well if taking a detailed look at the Differential Distribution Table (DDT) and the **BitPermutation**. The 1-round iterative differential characteristic $\varDelta W^i \rightarrow \varDelta W^{i+1}$ is

$$\begin{aligned}\varDelta W^i &= \texttt{0x0000 0000 0000 0000 0000 0400 0000 0000},\\ \varDelta W^{i+1} &= \texttt{0x0000 0000 0000 0000 0000 0400 0000 0000},\end{aligned}$$

which holds with probability of 2^{-3}. An illustration can be seen in Fig. 2.

The correctness of this differential can be seen as follows. Based on DDT, the following difference propagation $\varDelta W^i \rightarrow \varDelta W^i_s$ will hold with probability of 2^{-3}.

$$\begin{aligned}\varDelta W^i &= \texttt{0x0000 0000 0000 0000 0000 0400 0000 0000},\\ \varDelta W^i_s &= \texttt{0x0000 0000 0000 0000 0000 0200 0000 0000}.\end{aligned}$$

As can be observed, $\varDelta W^i_s[41] = 1$ and $\varDelta W^i_s[j] = 0$ $(0 \le j \le 127, j \ne 41)$. According to the definition of **BitPermutation** in Table 3, we have

$$\begin{aligned}&\varDelta W^i_p[42] = \varDelta W^i_s[41] = 1,\\ &\varDelta W^i_p[j] = 0\ (0 \le j \le 127, j \ne 42).\end{aligned}$$

Consequently,

$$\varDelta W^{i+1} = \varDelta W^i_p = \texttt{0x0000 0000 0000 0000 0000 0400 0000 0000}.$$

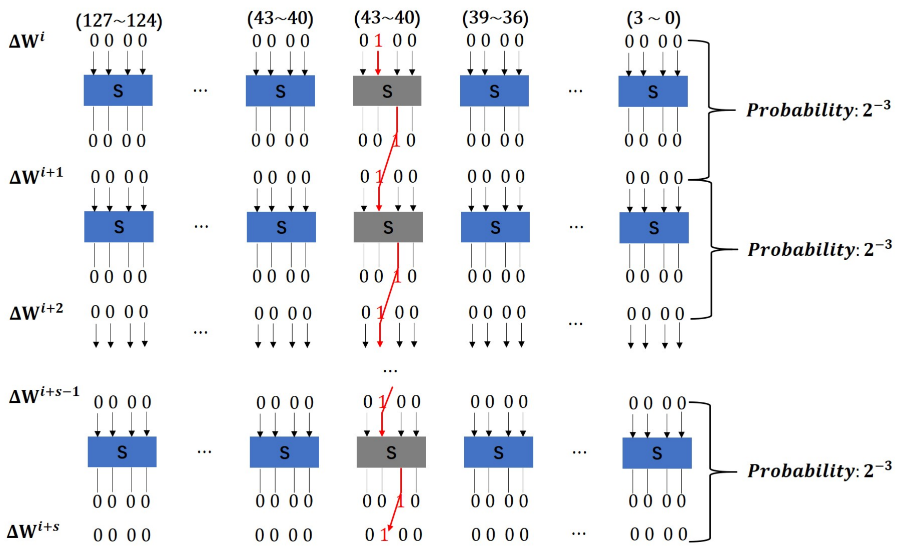


Fig. 2: The s-round iterative differential characteristic. The gray box represents the active s-box and the blue box represents the inactive s-box.

4 Application

With the 1-round iterative differential characteristic, we can mount distinguishing attack on reduced TRIFLE-BC, forgery attack on reduced TRIFLE and key-recovery attack on reduced TRIFLE. The details of the three attacks will be presented as follows.

4.1 Distinguishing Attack on Reduced TRIFLE-BC

Since there is a 1-round iterative differential characteristic holding with probability of 2^{-3}, an intuitive way is to construct an n-round distinguisher holding with probability of 3^{-3n}. Indeed, the distinguisher can be better. Instead of starting from the input difference $\varDelta_{it}$ and ending at the output difference $\varDelta_{it}$ for the n-round differential, where

$$\varDelta_{it} = \texttt{0x0000 0000 0000 0000 0000 0400 0000 0000},$$

we can start from the input difference $\varDelta_{st}$ and end at the output difference $\varDelta_{en}$ so that the difference propagation in the first round ($\varDelta_{st} \rightarrow \varDelta_{it}$) and the last round ($\varDelta_{it} \rightarrow \varDelta_{end}$) can hold with the highest probability. According to DDT, the choice for $\varDelta_{st}$ and $\varDelta_{en}$ is

$$\varDelta_{st} = \texttt{0x0000 0000 0000 0000 0000 0b00 0000 0000},$$
$$\varDelta_{en} = \texttt{0x0000 0000 0000 0000 0000 0400 0000 0400}.$$

The reason is that both $S(b \oplus x) \oplus S(x) = 2$ and $S(4 \oplus x) \oplus S(x) = 3$ ($x \in F_2^4$) hold with probability 2^{-2}. In other words, for the n-round differential characteristic $\varDelta_{st} \rightarrow \varDelta_{en}$, the iterative differential characteristic $\varDelta_{it} \rightarrow \varDelta_{it}$ is located in the intermediate $n-2$ rounds. As a result, the n-round differential characteristic $\varDelta_{st} \rightarrow \varDelta_{en}$ will hold with probability of 2^{2-3n}. With such an $n-$round differential characteristic, we can mount distinguishing attack on n-round TRIFLE-BC with data complexity 2^{3n-2} and time complexity 2^{3n-2}. Thus, the distinguishing attack can reach up to 42 (out of 50) rounds of TRIFLE-BC.

Indeed, since there is no whitening key used in TRIFLE-BC, we can peel off the **SubNibbles** and **BitPermutation** operations in the first round. In other words, we start the distinguishing attack from the state W_p^0. In this way, we can increase the above distinguisher by one more round. Thus, we can mount distinguishing attack on $(n+1)$-round TRIFLE-BC with data complexity 2^{3n-2} and time complexity 2^{3n-2}. Thus, the distinguishing attack can reach up to 43 (out of 50) rounds of TRIFLE-BC, whose time complexity and data complexity are both 2^{124}.

Remark The designers of TRIFLE [9] have used the MILP method [21] to find the best differential characteristics for up to 10 rounds and list their corresponding probability. The best differential characteristics for r ($4 \leq r \leq 10$) rounds all hold with probability of 2^{2-3r}. In fact, according to the DDT, we can observe that all the difference transactions ($1 \rightarrow 8$, $2 \rightarrow 1$, $4 \rightarrow 2$, $8 \rightarrow 4$) through an S-box hold with probability 2^{-3}. Since the linear transform is only a bit permutation, one can always construct 128 such differential characteristics, where only one bit is active in the input of each round, thus explaining why the solver returns such a result. Among all of them, the one-round iterative differential characteristic is unique and can be easily utilized to mount forgery attack, as will be detailed below.

4.2 Forgery Attack on Reduced TRIFLE

The overview of our forgery attack on n-round TRIFLE is illustrated in Fig. 3, where

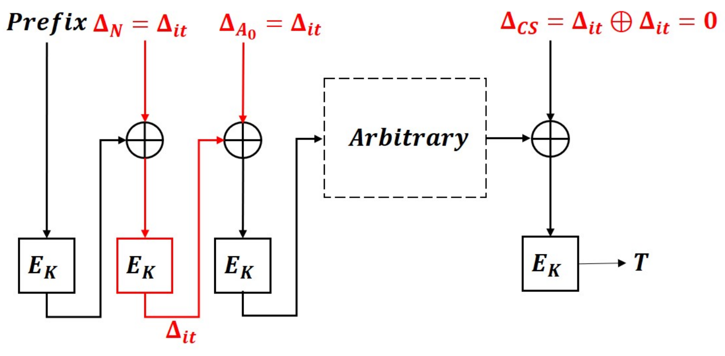
$$\varDelta_{it} = \texttt{0x0000 0000 0000 0000 0000 0400 0000 0000}.$$

Specifically, insert difference at the nonce N and the first associated data block A_0 so that $\varDelta_N = \varDelta_{it}$ and $\varDelta_{A_0} = \varDelta_{it}$. In this way, $\varDelta_{CS} = \varDelta_N \oplus \varDelta_{A_0} = 0$. Then, if the output difference of the second block equals to $\varDelta_{it}$, a collision of the tag T is found. Therefore, the success probability to mount the forgery attack on n-round TRIFLE is equal to the probability of the n-round differential characteristic $\varDelta_{it} \rightarrow \varDelta_{it}$, which holds with probability 2^{-3n}. The procedure of our forgery attack is as follows. Since the data complexity cannot exceed 2^{64}, we set the message M empty.

Step 1: The attacker randomly choose a nonce N and a 128-bit associated data A. Then, he sends an encryption query (A, M, N) and receives (C, T).

Step 2: The attacker then sends an decryption query $(A \oplus \varDelta_{it}, C, T, N \oplus \varDelta_{it})$. If the decryption succeeds, a forgery is achieved. Otherwise, return Step 1 until the decryption succeeds.

Therefore, we can mount forgery attack on TRIFLE for up to 21 rounds. The corresponding data complexity and time complexity are both 2^{63}. Note that a generic forgery attack on full TRIFLE will require 2^{64} queries. Therefore, our forgery attack on 21 rounds of TRIFLE only slightly outperforms the generic attack. For forgery attack on smaller rounds r ($1 \leq r \leq 20$), the data and time complexity are both 2^{3r}. Since the designers claim that TRIFLE is a nonce misuse resistant primitive, the same nonce can be reused. Thus, we can efficiently construct a forgery under the same master key with only one query after the above forgery attack procedure succeeds. Specifically, once we know (A,M,N) and $(A \oplus \varDelta_{it},M,N \oplus \varDelta_{it})$ can generate the same tag T, where M is empty, we can randomly choose another value for A' and send an encryption query (A', M, N) to receive (C', T'). Then, we can always know that $(A' \oplus \varDelta_{it}, C', T', N \oplus \varDelta_{it})$ is a valid forgery.



Prefix $\Delta_N = \Delta_{it}$
$\Delta_{A_0} = \Delta_{it}$
$\Delta_{CS} = \Delta_{it} \oplus \Delta_{it} = 0$
Arbitrary
E_K
E_K
E_K
E_K
T
Δ_{it}

Fig. 3: Overview of forgery attack

4.3 Key-recovery Attack on Reduced TRIFLE

In this section, we present a key-recovery attack on reduced TRIFLE under the classical differential attack framework. Similarly, we firstly give an overview of our attack in

Fig. 4. Different from the forgery attack, we expect there is difference in the input to the

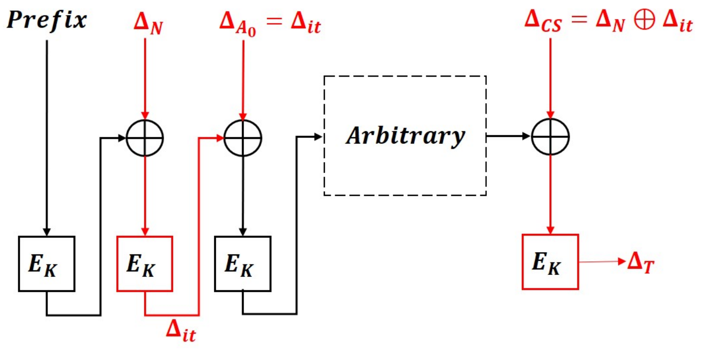

Prefix
Δ_N
$\Delta_{A_0} = \Delta_{it}$
$\Delta_{CS} = \Delta_N \oplus \Delta_{it}$
Arbitrary
E_K
E_K
E_K
E_K
Δ_T
Δ_{it}

Fig. 4: Overview of key-recovery attack

last block to generate the tag. Therefore, we set $\Delta_N \neq \Delta_{it}$ so that $\Delta_{CS} \neq 0$ and $\Delta_T \neq 0$. However, to make the best use of the iterative differential characteristic, we have to ensure that the 1-round differential characteristic $\Delta_N \rightarrow \Delta_{it}$ and $\Delta_N \oplus \Delta_{it} = \Delta_{CS} \rightarrow \Delta_{it}$ is possible. Actually, similar case has been discussed in the distinguishing attack. According to DDT, we set the choice for Δ_N as

$$\Delta_N = \texttt{0x0000 0000 0000 0000 0000 0f00 0000 0000}.$$

Then, Δ_{CS} will be

$$\Delta_{CS} = \texttt{0x0000 0000 0000 0000 0000 0b00 0000 0000}.$$

In this way, the 1-round differential characteristic $\Delta_N \rightarrow \Delta_{it}$ and $\Delta_{CS} \rightarrow \Delta_{it}$ will hold with probability of 2^{-3} and 2^{-2} respectively.

Suppose the aim is to mount key-recovery attack on r rounds of TRIFLE. Then, at the last block to generate the tag, we guess the last two round keys $(U^{r-1}, V^{r-1}, U^{r-2}, V^{r-2})$ and compute backward to observe the input difference of the two active S-boxes in the $(r-2)$-th round. In this case, the differential used for the key-recovery attack is equivalent to a $(2r-2)$-round differential $(\Delta_N \rightarrow \Delta_{en})$, as depicted in Fig. 5. The value of Δ_{en} is the same as defined in Section 4.2, as shown below.

$$\Delta_{en} = \texttt{0x0000 0000 0000 0000 0000 0400 0000 0400}.$$

Consequently, the equivalent $(2r-2)$-round differential $(\Delta_N \rightarrow \Delta_{en})$ will hold with probability of $2^{2-3(2r-2)}$. Since the data complexity cannot exceed 2^{64}, r can be at most 11. In other words, to mount key-recovery attack on 11-round TRIFLE, we can use an equivalent 20-round iterative differential characteristic as a distinguisher, which holds with probability 2^{-58}.

Now, we give a detailed description of the phase to recover the key with the equivalent 20-round differential $(\Delta_N \rightarrow \Delta_{en})$. According to Δ_{en}, we need to observe

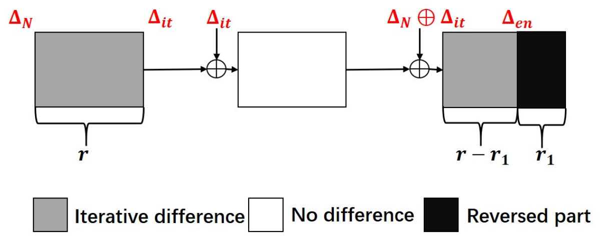


Fig. 5: Presentation of the differential used for key-recovery attack

the difference $\varDelta W^9[43 \sim 40]$ and $\varDelta W^9[11 \sim 8]$. Before stating the procedure to recover the key, we firstly explain which bits of the last two round key should be guessed. An illustration is given in Fig 6 and a detailed description is given below.

- To compute $W^9[43 \sim 40]$, we need to know $(W_p^9[106], W_p^9[74], W_p^9[42], W_p^9[10])$, which requires the knowledge of $(U^9[26], U^9[18], U^9[10], U^9[2])$.
- To compute $(W^{10}[106], W^{10}[74], W^{10}[42], W^{10}[10])$, we need to know $W_p^{10}[i]$, where

  $$i \in \{122, 90, 58, 26, 114, 82, 50, 18, 106, 74, 42, 10, 98, 66, 34, 2\}.$$

  Thus, we need to guess $U^{10}[j]$, where

  $$j \in \{30, 22, 14, 6, 28, 20, 12, 4, 26, 18, 10, 2, 24, 16, 8, 0\}.$$

- To compute $W^9[11 \sim 8]$, we need to know $(W_p^9[98], W_p^9[66], W_p^9[34], W_p^9[2])$, which requires the knowledge of $(U^9[24], U^9[16], U^9[8], U^9[0])$.
- To compute $(W^{10}[98], W^{10}[66], W^{10}[34], W^{10}[2])$, we need to know $W_p^{10}[i]$, where

  $$i \in \{120, 88, 56, 24, 112, 80, 48, 16, 104, 72, 40, 8, 96, 64, 32, 0\}.$$

  According to the definition of **AddRoundKey**, all the above 16 bits can be directly deduced from the value of tag. This is because that only partial bits of the internal states are xored with the round key.

As can be seen from the above explanation, we need to guess $16 + 8 = 24$ key bits in total in order to observe the difference $\varDelta W^9[43 \sim 40]$ and $\varDelta W^9[11 \sim 8]$. Now, we give the complete procedure to recover the key as follows.

Step 1: **Data Collection**. The attacker randomly chooses a nonce N and a 128-bit associated A and an empty message M. Then, he sends an encryption query (A, M, N) and receives (C, T). Then, he choose a new nonce $N' = N \oplus \varDelta_N$ and a new 128-bit associated $A' = A \oplus \varDelta_{it}$ and an empty message M. Then, he sends an encryption query (A', M, N') and receives (C', T'). He repeats this step 2^{t_0} times so as to obtain 2^{t_1} valid pairs (T, T') for key recovery phase. A valid pair (T, T') should satisfy that some bits of $\varDelta_T = T \oplus T'$ are zero, i.e., $\varDelta_T[j]$ should be zero, where

$$\begin{cases} 0 \le j \le 127 \\ j \notin \{122, 90, 58, 26, 114, 82, 50, 18, 106, 74, 42, 10, 98, 66, 34, 2\} \\ j \notin \{120, 88, 56, 24, 112, 80, 48, 16, 104, 72, 40, 8, 96, 64, 32, 0\} \end{cases}$$

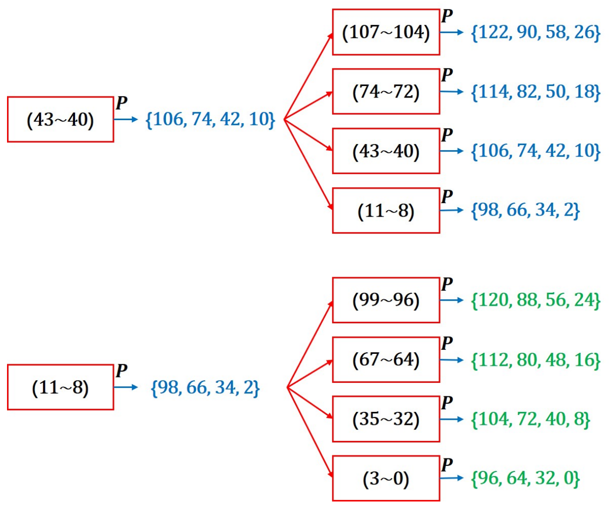


Fig. 6: To-be-guessed key bits in the last two rounds

Besides, since $W^{10}[i]$ are computable without knowing the key information, where

$$i \in \{120, 88, 56, 24, 112, 80, 48, 16, 104, 72, 40, 8, 96, 64, 32, 0\},$$

we need to check whether the difference ΔW^{10} is valid for the obtained (T, T'), i.e., among the above 16 bits, only $W^{10}[98]$,$W^{10}[66]$,$W^{10}[34]$ and $W^{10}[2]$ are allowed to have difference.

Step 2: **Key-Recovery**. The attacker initialize an array **CNT** of size 2^{24} with zero. Each row number of the array represents one possible value of the 24 to-be-guessed key bits. For the collected 2^{t_1} valid pairs (T, T'), the attacker guess the round key as follows.

Step 2.1: Independently guess the value of ($U^{10}[30]$,$U^{10}[22]$,$U^{10}[14]$,$U^{10}[6]$), ($U^{10}[28]$,$U^{10}[20]$,$U^{10}[12]$,$U^{10}[4]$), ($U^{10}[26]$,$U^{10}[18]$,$U^{10}[10]$,$U^{2}[4]$) and ($U^{10}[24]$,$U^{10}[16]$,$U^{10}[8]$,$U^{10}[0]$) and independently check whether the difference of $W^{10}[107 \sim 104]$, $W^{10}[74 \sim 72]$, $W^{10}[43 \sim 40]$ and $W^{10}[11 \sim 8]$ are valid difference, i.e., only $W^{10}[106]$,$W^{10}[74]$,$W^{10}[42]$ and $W^{10}[10]$ are allowed to have difference. If they are, record the corresponding guess. After this step, the attacker is expected to obtain about 2^4 possible values for the 16 guessed key bits of U^{10}. The time complexity of this step is $4 \times 2^4 = 2^6$. For each of the obtained possible values, the attacker carries out Step 2.2.

Step 2.2: The attacker independently guess the value of $U^9[i]$ ($i \in \{26, 18, 10, 2\}$) and $U^9[i]$ ($i \in \{24, 16, 8, 0\}$). Then, if the guessed value for $U^9[i]$ ($i \in \{26, 18, 10, 2\}$) and $U^9[i]$ ($i \in \{24, 16, 8, 0\}$) can make W^9 equal to Δ_{en}, the attacker accordingly increase the element by 1 in the row of **CNT** while represents the corresponding guessed value for the 24 to-be-guessed key bits.

Step 3: **Determine**. After all valid pairs (T, T') are used, check the array **CNT**. It is expected that the correct guess for the 24 to-be-guessed key bits will correspond to a row with maximum value.

Since the equivalent 20-round differential holds with probability 2^{-58} and the data complexity cannot exceed 2^{64}, we choose 2^{62} randomly values for (A, N) and expect to obtain 2^4 valid pairs (T, T') so that we can further confirm the correctness of the value for 24 to-be-guessed key bits. In this way, we can recover 24 bits of the key with time complexity and data complexity 2^{63}. The remaining 104 key bits can be guessed by brute force at the offline phase based on the key scheduling algorithm of TRIFLE-BC. Thus, the time complexity and data complexity of the key-recovery attack on 11-round TRIFLE are 2^{104} and 2^{63} respectively.

Key-Recovery Attack on TRIFLE-BC It is also interesting to investigate the key-recovery attack on TRIFLE-BC to help further understand the security of TRIFLE-BC, even though such an attack cannot work for real TRIFLE. Indeed the above key-recovery strategy can be completely applied to TRIFLE-BC trivially. Thus, we are able to mount key-recovery attack on TRIFLE-BC for up to 42+2=44 rounds by using the 42-round differential that holds with probability of 2^{-121}. We reduce the number of rounds of the distinguisher in Section 4.1 so that it can increase our confidence of the correctness of this attack. Similarly, we prepare $2^{121+4} = 2^{125}$ pairs of plaintext so that there are 2^4 valid ciphertext pairs for usage. Therefore, the total complexity of this key-recovery attack is dominated by the data collection phase, which requires 2^{126} plaintexts and 2^{126} time. We believe that such a key-recovery attack can be improved.

4.4 Experiments

To make the above theoretical results more convincing, we list a forgery (collision) for 10-round TRIFLE as following to demonstrate the correctness of the iterative differential characteristic. It is found with the reference implementation of TRIFLE [9]. To make the output in the implementation consistent with this paper, we changed the output format from the little-endian format to big-endian format.

$$\begin{aligned}
K &= \texttt{0x0a0a061117090f121b011a08030e0802},\\
N &= \texttt{0x1b050b100d130a06091900}\texttt{0415190e19},\\
A &= \texttt{0x061e1a0c04010410020216}\texttt{1a1b021a09},\\
M &= \text{Empty string},\\
T &= \texttt{0x28fa5821bf3278167acaa1b8f6de4603}.
\end{aligned}$$

$$\begin{aligned}
N \oplus \varDelta_{it} &= \texttt{0x1b050b100d130a06091904}\texttt{0415190e19},\\
A \oplus \varDelta_{it} &= \texttt{0x061e1a0c04010410020212}\texttt{1a1b021a09},\\
M &= \text{Empty string},\\
T &= \texttt{0x28fa5821bf3278167acaa1b8f6de4603}.
\end{aligned}$$

5 Conclusion

Although the designers claim that linear transform in TRIFLE-BC can provide sufficient diffusion, we can still find an iterative differential characteristic with hamming weight 1 by taking into account the combination of S-box and linear transform. With such a low-hamming-weight iterative differential characteristic, we can mount distinguishing attack on TRIFLE-BC for up to 43 rounds, forgery attack on TRIFLE for up to 21 rounds and key-recovery attack on TRIFLE for up to 11 rounds. We hope our result can help further understand the security of TRIFLE.

Acknowledgement We thank the anonymous reviewers of SAC 2019 and the TRIFLE team for their comments. According to the discussion with TRIFLE team, they are aware of such 1-1 differential transaction and therefore make such a transaction through an S-box hold with probability 2^{-3} rather than 2^{-2}. Fukang Liu is supported by Invitation Programs for Foreigner-based Researchers of the National Institute of Information and Communications Technology (NICT). Takanori Isobe is supported by Grant-in-Aid for Scientific Research (B) (KAKENHI 19H02141) for Japan Society for the Promotion of Science.

References

1. Roberto Avanzi. The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. *IACR Trans. Symmetric Cryptol.*, 2017(1):4–44, 2017.
2. Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 411–436, 2015.
3. Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 321–345, 2017.
4. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. SIMON and SPECK: block ciphers for the internet of things. *IACR Cryptology ePrint Archive*, 2015:585, 2015.
5. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 123–153, 2016.
6. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pages 2–21, 1990.
7. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight

block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, pages 450–466, 2007.

8. Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 272–288, 2009.
9. Nilanjan Datta, Ashrujit Ghoshal, Debdeep Mukhopadhyay, Sikhar Patranabis, Stjepan Picek, and Rajat Sadhukhan. TRIFLE, 2019. `https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates`.
10. Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 278–299, 2009.
11. Maria Eichlseder, Daniel Kales, and Markus Schofnegger. Forgery attacks on FlexAE and FlexAEAD. Cryptology ePrint Archive, Report 2019/679, 2019. `https://eprint.iacr.org/2019/679`.
12. Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 222–239, 2011.
13. Mustafa Khairallah. Forgery attack on SNEIKEN. Cryptology ePrint Archive, Report 2019/408, 2019. `https://eprint.iacr.org/2019/408`.
14. Lars R. Knudsen and David A. Wagner. Integral cryptanalysis. In *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, pages 112–127, 2002.
15. Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, pages 386–397, 1993.
16. Vasily Mikhalev, Frederik Armknecht, and Christian Müller. On ciphers that continuously access the non-volatile key. *IACR Trans. Symmetric Cryptol.*, 2016(2):52–79, 2016.
17. Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, pages 57–76, 2011.
18. Léo Perrin. Probability 1 iterated differential in the SNEIK permutation. Cryptology ePrint Archive, Report 2019/374, 2019. `https://eprint.iacr.org/2019/374`.
19. Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In *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 III*, pages 185–215, 2017.
20. Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 342–357, 2011.
21. Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 158–178, 2014.

22. Yosuke Todo. Structural evaluation by generalized integral property. In *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*, pages 287–314, 2015.
23. Yosuke Todo and Masakatu Morii. Bit-based division property and application to SIMON family. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 357–377, 2016.
24. Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 275–305, 2018.
25. 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 *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*, pages 648–678, 2016.