Bit-wise Cryptanalysis on AND-RX Permutation Friet-PC

Ryoma Ito[1], Rentaro Shiba[2], Kosei Sakamoto[2],
Fukang Liu[2], Takanori Isobe[1,2,3]

[1] National Institute of Information and Communications Technology, Japan.
`itorym@nict.go.jp`
[2] University of Hyogo, Japan.
`rentaro.shiba@gmail.com`,`k.sakamoto0728@gmail.com`,
`liufukangs@163.com`,`takanori.isobe@ai.u-hyogo.ac.jp`
[3] PRESTO, Japan Science and Technology Agency, Tokyo, Japan.

Abstract. This paper presents three attack vectors of bit-wise cryptanalysis including rotational, bit-wise differential, and zero-sum distinguishing attacks on the AND-RX permutation Friet-PC, which is implemented in a lightweight authenticated encryption scheme Friet. First, we propose a generic procedure for a rotational attack on AND-RX cipher with round constants. By applying the proposed attack to Friet-PC, we can construct an 8-round rotational distinguisher with a time complexity of 2^{102}. Next, we explore single- and dual-bit differential biases, which are inspired by the existing study on Salsa and ChaCha, and observe the best bit-wise differential bias with $2^{-9.552}$. This bias allows us to practically construct a 9-round bit-wise differential distinguisher with a time complexity of $2^{20.044}$. Finally, we construct 13-, 15-, 17-, and 30-round zero-sum distinguishers with time complexities of 2^{31}, 2^{63}, 2^{127}, and 2^{383}, respectively. To summarize our study, we apply three attack vectors of bit-wise cryptanalysis to Friet-PC and show their superiority as effective attacks on AND-RX ciphers.

Keywords: Authenticated Encryption · Permutation · Friet-PC · Rotational Attack · Bit-wise Differential Attack · Zero-sum Distinguisher

1 Introduction

1.1 Background

Friet, which was proposed by Simon et al. at EUROCRYPT 2020 [26], is a lightweight authenticated encryption scheme with a 128-bit security level that is resistant to side channel and fault injection attacks. It adopts the authenticated encryption mode SpongeWrap based on the duplex construction [5]. The SpongeWrap mode is based on the concept of efficiently building an authenticated encryption scheme from cryptographic permutation; thus, designers who adopt SpongeWrap as the authenticated encryption mode have an important task of designing a lightweight cryptographic permutation with a high security level. The designers of Friet proposed a new design technique for ciphers with efficient fault-detecting implementations, and then designed new cryptographic permutations called Friet-PC and Friet-P for implementation in Friet.

A previous version of the Friet-PC permutation, called Frit, was proposed by the same designers in 2018 [25]. It adopts the AND-Rotation-XOR (AND-RX) construction, which is much similar to the Addition-Rotation-XOR (ARX) construction. Shortly thereafter, Dobraunig et al. performed a key recovery attack against the full-round version in the use case of Frit as an Even-Mansour block cipher [9]. In addition, Qin et al. applied a cube attack on the reduced-round version in the use case of Frit as a duplex-based authenticated encryption mode [23]. Friet-PC was designed considering these attacks.

The designers evaluated the security of Friet-PC against differential and linear attacks [26]. They first investigated the propagation properties to determine the minimum weights of differential and linear trails, and then experimentally obtained a 6-round differential trail with weight 59 and an 8-round linear trail with weight 80. These trails can be extended to a 6-round differential distinguisher with a time complexity of 2^{59} and an 8-round linear distinguisher with a time complexity of 2^{80}. As a security evaluation by a third party, Liu et al. proposed a new framework called a rotational differential-linear attack [19], which is inspired from the differential-linear attack proposed by Langford and Hellman [17]. Their proposed attack significantly improved the security evaluation by the designers, and allowed us to construct a 13-round rotational differential-linear distinguisher with a time complexity of $2^{117.81}$. To the best of our knowledge, the security evaluation for Friet-PC by a third party has not been reported except for that by Liu et al.; thus, the best attack on Friet-PC is the 13-round rotational differential-linear distinguisher.

1.2 Our Contribution

In this study, we evaluate the security of Friet-PC with three attack vectors of bit-wise cryptanalysis: rotational, bit-wise differential, and zero-sum distinguishing attacks. Although these vectors are widely used as generic attacks against ARX and AND-RX ciphers, no study appears to have applied these attacks to evaluate the security of Friet-PC as yet. If an adversary can efficiently perform these attacks on Friet-PC, they may threaten the security of not only the permutation Friet-PC but also the authenticated encryption scheme Friet.

Table 1 summarizes the results of previous security evaluations and the evaluation in this study for Friet-PC. The proposed security evaluations sufficiently improve the existing best attack by Liu et al.; thus, we show their superiority as effective attacks on AND-RX ciphers. We remark that the proposed attacks are no practical threat to Friet-PC, however, it is recommended to use these attack vectors of bit-wise cryptanalysis to evaluate the security of AND-RX ciphers when designing the AND-RX ciphers in the future. The details of the proposed security evaluations are given in the following text.

Table 1. Summary of our results.

Attack type	Rounds	Time	Reference
Differential/Distinguisher	6	$2^{59.00}$	[26]
Linear/Distinguisher	8	$2^{80.00}$	[26]
Rotational Differential-Linear/Distinguisher	8	$2^{17.81}$	[19]
Rotational Differential-Linear/Distinguisher	9	$2^{29.81}$	[19]
Rotational Differential-Linear/Distinguisher	13	$2^{117.81}$	[19]
Rotational/Distinguisher	8	$2^{102.00}$	Section 3
Bit-wise Differential/Distinguisher	9	$2^{20.04}$	Section 4
Zero-sum/Distinguisher	13	$2^{31.00}$	Section 5
Zero-sum/Distinguisher	15	$2^{63.00}$	Section 5
Zero-sum/Distinguisher	17	$2^{127.00}$	Section 5
Zero-sum/Distinguisher	30	$2^{383.00}$	Section 5

Rotational Attack We perform a rotational attack, which has been mainly applied to ARX and AND-RX ciphers [1, 4, 10, 12, 13, 14, 15, 16, 18, 20, 22]. We propose a new technique called XOR masking technique, and generalize an attack procedure for a rotational attack on AND-RX ciphers with round constants. No existing attacks are known to use the concept of masking specific values for the input/output values of the target ciphers as done in the proposed technique. To apply the proposed XOR masking technique to the target ciphers, we analyze the influence of round constants on the XOR and AND operations in detail and obtain appropriate mask values for the input/output values of Friet-PC. By applying the generalized attack procedure with the proposed XOR masking technique to the reduced-round Friet-PC, an 8-round rotational distinguisher with a time complexity of 2^{102} was achieved. We believe that it is feasible to use the proposed attack procedure against other ARX and AND-RX ciphers with round constants.

Bit-wise Differential Attack We conduct a bit-wise differential attack, which has been mainly applied to ARX stream ciphers [2, 6, 24]. We experimentally explore single- and dual-bit differential biases of the 9-, 10-, and 11-round Friet-PC. By using the best dual-bit differential bias of the 9-round Friet-PC, we demonstrate a practical bit-wise differential distinguisher for the 9-round Friet-PC with a time complexity of $2^{20.044}$. The existing bit-wise differential attack has been applied only to ARX ciphers so far, however, it can be clarified that this attack is also effective for AND-RX ciphers in this study.

Zero-sum Distinguisher We compute the bound of the algebraic degree for a certain number of rounds of Friet-PC with the bit-based division property [27]. By precisely modeling each operation of Friet-PC and carefully choosing

Algorithm 1 FRIET-PC

```
Input: $a, b, c \in \{0,1\}^{128}$
Output: $(a', b', c') \leftarrow$ FRIET-PC$(a, b, c)$
 1: procedure FRIET-PC$(a, b, c)$
 2:     for $i = 0$ to 23 do
 3:         $c \leftarrow c \oplus \mathsf{rc}_i$                                  $\triangleright\ \delta_i$
 4:         $(a, b, c) \leftarrow (a \oplus b \oplus c, c, a)$                    $\triangleright\ \tau_1$
 5:         $b \leftarrow b \oplus (c \lll 1)$                                 $\triangleright\ \mu_1$
 6:         $c \leftarrow c \oplus (b \lll 80)$                                $\triangleright\ \mu_2$
 7:         $(a, b, c) \leftarrow (a, a \oplus b \oplus c, c)$                    $\triangleright\ \tau_2$
 8:         $a \leftarrow a \oplus ((b \lll 36) \wedge (c \lll 67))$                $\triangleright\ \xi$
 9:     end for
10: return $(a, b, c)$
11: end procedure
```

input patterns, we succeed in constructing 13-, 15-, 17-, and 30-round zero-sum distinguishers [3] with time complexities of 2^{31}, 2^{63}, 2^{127}, and 2^{383}, respectively. To the best of our knowledge, these are the best distinguishers for reduced-round FRIET-PC, given that the attacker has a full control over the internal state, which is a common assumption to analyze the security of a public permutation.

1.3 Organization of the Paper

The rest of the paper is organized as follows. In Section 2, we briefly describe the specification of the FRIET-PC permutation. In Section 3, we first review the existing techniques for the rotational attacks, and propose a generic attack procedure for a rotational attack on AND-RX ciphers with round constants. Based on the proposed attack procedure, we provide a rotational distinguisher for the 8-round FRIET-PC. In Section 4, we first introduce the existing techniques for the bit-wise differential attacks, and then provide a bit-wise differential distinguisher for the 9-round FRIET-PC. In Section 5, we first describe the how to search for integral distinguishers with the bit-based division property, and then provide the zero-sum distinguishers for the 13-, 15-, 17-, and 30-round FRIET-PC. Finally, Section 6 concludes the paper.

2 Specifications of FRIET-PC Permutation

FRIET-PC has three limbs $(a, b, c) \in \{0,1\}^{128}$, and its round function consists of the following six steps:

- a round constant addition step δ_i that is a limb adaptation,
- two non-native limb transposition steps τ_1 and τ_2,
- two mixing steps μ_1 and μ_2 that are limb adaptations, and
- a nonlinear step ξ that is also a limb adaptation.

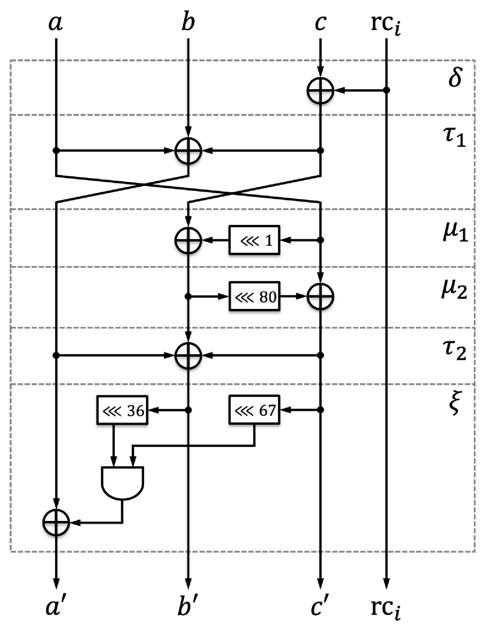

Fig. 1. Round function of Friet-PC.

Table 2. Round constants rc_i in hexadecimal notation, omitting the leading zero digits.

i	rc_i	i	rc_i	i	rc_i	i	rc_i	i	rc_i	i	rc_i
0	1111	4	101	8	1001	12	1	16	1110	20	1011
1	11100000	5	10110000	9	100000	13	110000	17	11010000	21	1100000
2	1101	6	110	10	100	14	111	18	1010	22	1100
3	10100000	7	11000000	11	10000000	15	11110000	19	1010000	23	10010000

We describe the procedure of the Friet-PC permutation as shown in Algorithm 1 and Fig. 1, and use the following notation for this procedure:

– $x \oplus y$ is the exclusive *or* (XOR) of two limbs x and y,
– $x \wedge y$ is the bit-wise logical *and* (AND) of two limbs x and y,
– $x \lll n$ is the left rotation by n bits of a limb x, and
– rc_i is the i-th round constant as listed in Table 2.

We use this notation throughout the remainder of this paper.

3 Rotational Distinguisher

We analyze the security of Friet-PC against a rotational attack, which has been applied to ARX and AND-RX ciphers, such as block ciphers Threefish [13], Speck [1, 18], Simon [20] and Simeck [20]; stream ciphers Salsa [12] and ChaCha

[4]; hash functions Keccak [22], BLAKE2 [10, 14] and Skein [14, 15]; and message authentication code algorithm Chaskey [16]. In this section, we first review the generic techniques for the rotational attacks and subsequently explain a new technique for a rotational attack on AND-RX ciphers with round constants. Then, we describe the application of the proposed technique to FRIET-PC and finally show a rotational distinguisher for the 8-round FRIET-PC with a time complexity of 2^{102}.

3.1 Rotational Attacks

In 2010, Khovratovich and Nikolić [13] explored the propagation of a rotational pair $(X, X \lll r)$ or $(X, X \ggg r)$ throughout an ARX cipher, and generalized a new technique called rotational attack. In the following text, we discuss only the propagation of the rotation pair $(X, X \lll r)$, as the propagation of the rotation pair $(X, X \ggg r)$ can be explained similarly. A rotational attack on an ARX or AND-RX cipher allows an adversary to analyze the rotational probability of the entire cipher by multiplying the individual rotational probabilities of all operations used in the cipher. In other words, the adversary can properly perform the rotational attack on an ARX or AND-RX cipher by computing the rotational probabilities of four distinct operations, i.e., modular addition, AND, rotation, and XOR. The rotational probabilities of AND, rotation, and XOR are given by

$$\Pr[(X \wedge Y) \lll r = (X \lll r) \wedge (Y \lll r)] = 1, \tag{1}$$
$$\Pr[(X \lll r_1) \lll r_2 = (X \lll r_2) \lll r_1] = 1, \tag{2}$$
$$\Pr[(X \oplus Y) \lll r = (X \lll r) \oplus (Y \lll r)] = 1, \tag{3}$$

while the rotational probability of modular addition is given by the following lemma.

Lemma 1 ([8, Corollary 4.12]). *If we suppose an n-bit word X to be fixed and an n-bit word Y to be chosen uniformly at random, then we obtain*

$$\Pr[(X + Y) \lll r = (X \lll r) + (Y \lll r)] = 2^{-n}(2^{n-r} - X_R)(2^r - X_L), \tag{4}$$

where $X_L = (x_{n-1}, \ldots, x_{n-r})$ and $X_R = (x_{n-r-1}, \ldots, x_0)$ for X.

On the other hand, if we suppose two n-bit words X and Y to be chosen uniformly at random, then we obtain

$$\Pr[(X + Y) \lll r = (X \lll r) + (Y \lll r)] = \frac{1}{4}(1 + 2^{r-n} + 2^{-r} + 2^{-n}). \tag{5}$$

It should be noted here that **all** inputs to an ARX or AND-RX must be rotational pair for the rotational attack to perform well, claimed by Khovratovich and Nikolić [13]. According to them, we cannot perform a proper rotational attack on an ARX or AND-RX cipher with round constants such as FRIET-PC, because it is practically difficult to obtain a rotational pair of round constants. To solve this problem, some studies explored a rotational attack against ARX and AND-RX block ciphers Speck [1, 18] and Simon [20] with constants that actually correspond to round keys. However, no study on a rotational attack against an ARX or AND-RX cipher with round constants specified in the specification, such as Friet-PC, has been reported as yet.

3.2 Rotational Attack on AND-RX Ciphers with Round Constants

To properly perform a rotational attack on an AND-RX cipher with round constants, we first demonstrate that the XOR operation in the presence of round constants can preserve the propagation of a rotational pair with a probability of one by introducing a XOR masking technique into a rotational attack. Then, we establish the rotational probability of the AND operation in the presence of round constants. Finally, we propose a generic attack procedure for a rotational attack on AND-RX ciphers with round constants. In the following text, we describe a rotational pair as $(X, \overleftarrow{X})$ instead of $(X, X \lll r)$.

XOR Masking Technique for the XOR Operation with Constants. We first introduce a XOR masking technique so that the XOR operation in the presence of round constants rc expressed in the form

$$\overleftarrow{X \oplus rc} \overset{?}{=} \overleftarrow{X} \oplus rc \oplus mask_1 \tag{6}$$

satisfies the equality. The left side of (6) is not XOR masked as it satisfies the same form as the left side of Eq.(3). Then, it can be seen from Eq.(3) that (6) satisfies the equality with a probability of one when $mask_1 = rc \oplus \overleftarrow{rc}$ because

$$\begin{aligned} \overleftarrow{X \oplus rc} &= \overleftarrow{X} \oplus rc \oplus mask_1 \\ &= \overleftarrow{X} \oplus rc \oplus rc \oplus \overleftarrow{rc} \\ &= \overleftarrow{X} \oplus \overleftarrow{rc}. \end{aligned} \tag{7}$$

In summary, the XOR operation in the presence of round constants can preserve the propagation of a rotational pair with a probability of one by XORing the mask value $mask_1 = rc \oplus \overleftarrow{rc}$. Note that the XOR masking technique can be applied to both the input and output values of the target cipher. For example, when the adversary applies the XOR masking technique to the input value, he/she must choose $(X, \overleftarrow{X} \oplus mask_1)$ as the input rotational pair.

XOR Masking Technique for the AND Operation with Constants. We examine whether the AND operation in the presence of a round constants, rc_1 and rc_2, expressed in the form

$$\overleftarrow{(X \oplus rc_1) \wedge (Y \oplus rc_2)} \overset{?}{=} (\overleftarrow{X} \oplus rc_1) \wedge (\overleftarrow{Y} \oplus rc_2) \tag{8}$$

satisfies the equality. To reveal the differences between both sides of (8), we use Eqs.(1) and (3) to transform (8) to

$$(\overleftarrow{X} \oplus \overleftarrow{rc_1}) \wedge (\overleftarrow{Y} \oplus \overleftarrow{rc_2}) \overset{?}{=} (\overleftarrow{X} \oplus rc_1) \wedge (\overleftarrow{Y} \oplus rc_2). \tag{9}$$

Table 3. Truth table for Example 1.

$\overleftarrow{x_i} \oplus \overleftarrow{rc_{1,i}}$	$\overleftarrow{x_i} \oplus rc_{1,i}$	$\overleftarrow{y_i} \oplus \overleftarrow{rc_{2,i}}$	$\overleftarrow{y_i} \oplus rc_{2,i}$	$(\overleftarrow{x_i} \oplus \overleftarrow{rc_{1,i}}) \wedge (\overleftarrow{y_i} \oplus \overleftarrow{rc_{2,i}})$	$(\overleftarrow{x_i} \oplus rc_{1,i}) \wedge (\overleftarrow{y_i} \oplus rc_{2,i})$
0	1	0	0	**0**	**0**
		1	1	0	1
1	0	0	0	**0**	**0**
		1	1	1	0

We then apply the XOR masking technique to the input value so that the AND operation in the presence of round constants expressed in the form

$$(\overleftarrow{X} \oplus \overleftarrow{rc_1}) \wedge (\overleftarrow{Y} \oplus \overleftarrow{rc_2}) \overset{?}{=} (\overleftarrow{X} \oplus rc_1 \oplus mask_2) \wedge (\overleftarrow{Y} \oplus rc_2 \oplus mask_3) \tag{10}$$

satisfies the equality. Here, (10) satisfies the equality with a probability of one when $(mask_2, mask_3) = (rc_1 \oplus \overleftarrow{rc_1}, rc_2 \oplus \overleftarrow{rc_2})$ because

$$\begin{aligned}(\overleftarrow{X} \oplus \overleftarrow{rc_1}) \wedge (\overleftarrow{Y} \oplus \overleftarrow{rc_2}) &= (\overleftarrow{X} \oplus rc_1 \oplus mask_2) \wedge (\overleftarrow{Y} \oplus rc_2 \oplus mask_3)\\ &= (\overleftarrow{X} \oplus rc_1 \oplus rc_1 \oplus \overleftarrow{rc_1}) \wedge (\overleftarrow{Y} \oplus rc_2 \oplus rc_2 \oplus \overleftarrow{rc_2})\\ &= (\overleftarrow{X} \oplus \overleftarrow{rc_1}) \wedge (\overleftarrow{Y} \oplus \overleftarrow{rc_2}) \end{aligned} \tag{11}$$

This implies that the adversary must choose $[(X, \overleftarrow{X} \oplus mask_2), (Y, \overleftarrow{Y} \oplus mask_3)]$ as the input rotational pair when he/she applies the XOR masking technique to the input value.

Similarly, we apply the XOR masking technique to the output value corresponding to the input rotational pairs so that the AND operation in the presence of round constants expressed in the form

$$[(\overleftarrow{X} \oplus \overleftarrow{rc_1}) \wedge (\overleftarrow{Y} \oplus \overleftarrow{rc_2})] \oplus mask_4 \overset{?}{=} [(\overleftarrow{X} \oplus rc_1) \wedge (\overleftarrow{Y} \oplus rc_2)] \oplus mask_5 \tag{12}$$

satisfies the equality. However, it is practically difficult to determine the appropriate mask values so that (12) satisfies the equality. We will explain the reason after providing the following two examples. Let x_i, y_i, $rc_{1,i}$, and $rc_{2,i}$ be the i-th bit of X, Y, rc_1, and rc_2, respectively.

Example 1. We focus on the AND operation of the i-th bit in (9). We assume that either $rc_{1,i} \oplus \overleftarrow{rc_{1,i}} = 1$ or $rc_{2,i} \oplus \overleftarrow{rc_{2,i}} = 1$ holds. In this example, we assume that $rc_{1,i} \oplus \overleftarrow{rc_{1,i}} = 1$ holds for the sake of simplicity. Table 3 provides a truth table corresponding to (9). This table shows that the AND operation of the i-th bit holds with a probability of 2^{-1}.

Example 2. We also focus on the AND operation of the i-th bit in (9). In this example, we assume that both $rc_{1,i} \oplus \overleftarrow{rc_{1,i}} = 1$ and $rc_{2,i} \oplus \overleftarrow{rc_{2,i}} = 1$ hold. Table 4 provides a truth table corresponding to (9). This table shows that the AND operation of the i-th bit holds with a probability of 2^{-1}.

Table 4. Truth table for Example 2.

$\overleftarrow{x_i} \oplus \overleftarrow{rc_{1,i}}$	$\overleftarrow{x_i} \oplus rc_{1,i}$	$\overleftarrow{y_i} \oplus \overleftarrow{rc_{2,i}}$	$\overleftarrow{y_i} \oplus rc_{2,i}$	$(\overleftarrow{x_i} \oplus \overleftarrow{rc_{1,i}}) \wedge (\overleftarrow{y_i} \oplus \overleftarrow{rc_{2,i}})$	$(\overleftarrow{x_i} \oplus rc_{1,i}) \wedge (\overleftarrow{y_i} \oplus rc_{2,i})$
0	1	0	1	0	1
		1	0	**0**	**0**
1	0	0	1	**0**	**0**
		1	0	1	0

These examples show that the AND operation of the i-th bit in (9) holds with a probability of 2^{-1} when at least either $rc_{1,i} \oplus \overleftarrow{rc_{1,i}} = 1$ or $rc_{2,i} \oplus \overleftarrow{rc_{2,i}} = 1$ holds. Moreover, these examples show bitwise independent events since (9) is a bit-wise operation; thus, we can compute a probability that the AND operation expressed in (9) satisfies the equality by simply counting the number of bits for which either $rc_{1,i} \oplus \overleftarrow{rc_{1,i}} = 1$ or $rc_{2,i} \oplus \overleftarrow{rc_{2,i}} = 1$ holds for each bit. These facts lead to the following theorem.

Theorem 1. *Let $(X, \overleftarrow{X})$ and $(Y, \overleftarrow{Y})$ be two rotational pairs where symbol '$\leftarrow$' represents the left rotation by r bits, and let rc_1 and rc_2 be round constants. Then, the rotational probability of the AND operation in the presence of round constants is given as follows:*

$$\Pr\left[\overleftarrow{(X \oplus rc_1) \wedge (Y \oplus rc_2)} = (\overleftarrow{X} \oplus rc_1) \wedge (\overleftarrow{Y} \oplus rc_2)\right] = 2^{-hw[(rc_1 \oplus \overleftarrow{rc_1}) | (rc_2 \oplus \overleftarrow{rc_2})]}, \tag{13}$$

where $hw[\cdot]$ represents the hamming weight.

Proof. As discussed earlier, the AND operation of the i-th bit in (9) holds with a probability of 2^{-1} when at least either $rc_{1,i} \oplus \overleftarrow{rc_{1,i}} = 1$ or $rc_{2,i} \oplus \overleftarrow{rc_{2,i}} = 1$ holds. Moreover, we can compute a probability that the AND operation expressed in (9) satisfies the equality by simply counting the number of bits for which either $rc_{1,i} \oplus \overleftarrow{rc_{1,i}} = 1$ or $rc_{2,i} \oplus \overleftarrow{rc_{2,i}} = 1$ holds for each bit. We can achieve this by calculating the hamming weight such as $hw[(rc_1 \oplus \overleftarrow{rc_1}) \mid (rc_2 \oplus \overleftarrow{rc_2})]$. In summary, the rotational probability of the AND operation in the presence of round constants is given as shown in Eq.(13).

Now, we explain why it is practically difficult to determine the appropriate mask values so that (12) satisfies the equality. This is because the mask values cannot be uniquely determined unless the adversary knows the correct values of X and Y, which are usually the intermediate information of the target cipher and are not available to the adversary (with exceptions). For example, from Table 3, if the values of $\overleftarrow{y_i} \oplus \overleftarrow{rc_{2,i}}$ and $\overleftarrow{y_i} \oplus rc_{2,i}$ are 1, the adversary must apply the XOR mask to satisfy the equality, but he/she cannot decide whether to apply the XOR mask without knowing the value of $\overleftarrow{y_i}$. Therefore, we should evaluate a rotational probability of the AND operation in the presence of round constants according to Theorem 1 without applying the XOR masking technique to the output value corresponding to the input rotational pairs.

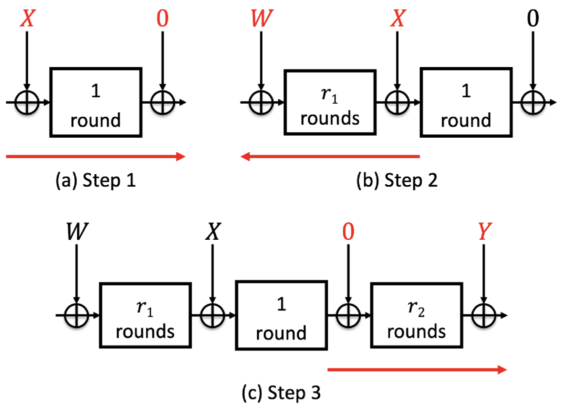

Fig. 2. Proposed attack procedure: (a) Step 1, (b) Step 2, and (c) Step 3.

Attack Procedure. Based on the discussed XOR masking technique, we propose a generic attack procedure for a rotational attack on AND-RX ciphers with round constants. The proposed attack consists of offline and online phases. In the offline phase, we perform the following procedure:

Step 1. We analyze the input and output mask values for the i-th round function of the target AND-RX cipher. In this step, we apply the XOR masking technique to the input rotational pair so that the influence of the round constant does not propagate to the output rotational pair. As shown in Fig. 2 (a), the input rotational pair is masked with a specific value X to cancel the influence of the round constant; then, we do not need to apply the XOR masking technique to the output rotational pair.

Step 2. We explore the input mask value for the $(i - r_1)$-th round function of the target AND-RX cipher by going back r_1 rounds from the i-th round function of the cipher. This is feasible because we can easily construct the inverse function of the AND-RX cipher. As shown in Fig. 2 (b), we obtain the input mask value W for the $(i - r_1)$-th round function such that the output mask value of the $(i - 1)$-th round function becomes X.

Step 3. We investigate the output mask value for the $(i+r_2)$-th round function of the target AND-RX cipher. As shown in Fig. 2 (c), the input mask value of the $(i+1)$-th round function is 0, as obtained in Step 1; then, we can obtain the output mask value Y for the $(i + r_2)$-th round function by analyzing the influence of the round constants through the r_2 rounds of the target AND-RX cipher.

We finally obtain the input mask value W and the output mask value Y for the (r_1+r_2+1)-round version of the target AND-RX cipher. Thereafter, in the online phase, by utilizing these mask values, we can construct a rotational distinguisher for the target AND-RX cipher in a manner similar to that in existing studies [1, 4, 10, 12, 13, 14, 15, 16, 18, 20, 22].

3.3 Application to Friet-PC

We apply the proposed attack procedure to Friet-PC. We first perform the offline phase of the proposed attack procedure on Friet-PC and obtain the input/output mask values for each round. Then, we examine the techniques for mitigating the influence of the round constants. Finally, we perform the online phase of the proposed attack procedure on Friet-PC, and demonstrate a rotational distinguisher for the 8-round Friet-PC with a time complexity of 2^{102}.

Offline Phase. Let $(mask_a^{(r)}, mask_b^{(r)}, mask_c^{(r)})$ be the input mask variables for the r-round limbs (a, b, c), or the output mask variables for the $(r-1)$-round limbs (a, b, c), respectively; let $\mathsf{RC}_r^{\lll t} = \mathsf{rc}_r^{\lll t} \oplus \overleftarrow{\mathsf{rc}_r^{\lll t}}$, where $\mathsf{rc}_r^{\lll t}$ is the r-th round constant with t-bit left rotation, and $(\mathsf{rc}_r^{\lll t}, \overleftarrow{\mathsf{rc}_r^{\lll t}})$ is a rotational pair of the r-th round constant.

Based on the offline phase in the proposed attack procedure, we obtain the input/output mask values for each round of Friet-PC as follows:

Step 1. We need to mask the i-round input rotational pair with a specific value to cancel the influence of the round constant. Algorithm 1 shows that the round constant rc_i is used for the first operation in the round function of Friet-PC, such as $c \leftarrow c \oplus \mathsf{rc}_i$; thus, we can obtain the i-round input/output mask values $(mask_a^{(i)}, mask_b^{(i)}, mask_c^{(i)}) = (0, 0, \mathsf{RC}_i^{\lll 0})$ and $(mask_a^{(i+1)}, mask_b^{(i+1)}, mask_c^{(i+1)}) = (0, 0, 0)$ because

$$\begin{aligned} \overleftarrow{c \oplus \mathsf{rc}_i^{\lll 0}} &= \overleftarrow{c} \oplus \mathsf{rc}_i^{\lll 0} \oplus mask_c^{(i)} \\ &= \overleftarrow{c} \oplus \mathsf{rc}_i^{\lll 0} \oplus \mathsf{RC}_i^{\lll 0} \\ &= \overleftarrow{c} \oplus \mathsf{rc}_i^{\lll 0} \oplus \mathsf{rc}_i^{\lll 0} \oplus \overleftarrow{\mathsf{rc}_i^{\lll 0}} \\ &= \overleftarrow{c} \oplus \overleftarrow{\mathsf{rc}_i^{\lll 0}} \end{aligned} \tag{14}$$

holds with a probability of one. The influence of the round constant is cancelled completely by using these input mask values.

Step 2. We need to mask the $(i-r_1)$-round input rotational pair with a specific value such that the output mask value of the $(i-1)$-th round function becomes $(mask_a^{(i)}, mask_b^{(i)}, mask_c^{(i)}) = (0, 0, \mathsf{RC}_i^{\lll 0})$; thus, by going back r_1 rounds from the i-th round function of Friet-PC, we can obtain the $(i-r_1)$-round input mask value. Table 5 lists the input mask values by going back up to $(i-3)$ rounds.

Step 3. In Step 1, we have obtained the i-round output mask value $(mask_a^{(i+1)}, mask_b^{(i+1)}, mask_c^{(i+1)}) = (0, 0, 0)$, which is the $(i+1)$-round input mask value. Thus, by analyzing the influence of the round constants through the r_2 rounds of Friet-PC, we can obtain the $(i+r_2)$-round output mask values. Table 5 lists the output mask values by going up to $(i+4)$ rounds.

Table 5. List of input/output mask values.

Mask variables	Input/output mask values
$mask_a^{(i-3)}$	$\mathsf{RC}_{i-2}^{\lll 0} \oplus \mathsf{RC}_{i-2}^{\lll 80} \oplus \mathsf{RC}_{i-1}^{\lll 0} \oplus \mathsf{RC}_{i-1}^{\lll 1} \oplus \mathsf{RC}_{i-1}^{\lll 81} \oplus \mathsf{RC}_{i}^{\lll 1} \oplus \mathsf{RC}_{i}^{\lll 2} \oplus \mathsf{RC}_{i}^{\lll 80} \oplus \mathsf{RC}_{i}^{\lll 82} \oplus \mathsf{RC}_{i}^{\lll 160}$
$mask_b^{(i-3)}$	$\mathsf{RC}_{i-2}^{\lll 1} \oplus \mathsf{RC}_{i-2}^{\lll 80} \oplus \mathsf{RC}_{i-2}^{\lll 81} \oplus \mathsf{RC}_{i-1}^{\lll 2} \oplus \mathsf{RC}_{i-1}^{\lll 80} \oplus \mathsf{RC}_{i-1}^{\lll 81} \oplus \mathsf{RC}_{i-1}^{\lll 82} \oplus \mathsf{RC}_{i}^{\lll 3} \oplus \mathsf{RC}_{i}^{\lll 82} \oplus \mathsf{RC}_{i}^{\lll 83} \oplus \mathsf{RC}_{i}^{\lll 160} \oplus \mathsf{RC}_{i}^{\lll 161}$
$mask_c^{(i-3)}$	$\mathsf{RC}_{i-3}^{\lll 0} \oplus \mathsf{RC}_{i-2}^{\lll 0} \oplus \mathsf{RC}_{i-2}^{\lll 1} \oplus \mathsf{RC}_{i-2}^{\lll 81} \oplus \mathsf{RC}_{i-1}^{\lll 1} \oplus \mathsf{RC}_{i-1}^{\lll 2} \oplus \mathsf{RC}_{i-1}^{\lll 82} \oplus \mathsf{RC}_{i}^{\lll 0} \oplus \mathsf{RC}_{i}^{\lll 2} \oplus \mathsf{RC}_{i}^{\lll 3} \oplus \mathsf{RC}_{i}^{\lll 80} \oplus \mathsf{RC}_{i}^{\lll 81} \oplus \mathsf{RC}_{i}^{\lll 83} \oplus \mathsf{RC}_{i}^{\lll 161}$
$mask_a^{(i-2)}$	$\mathsf{RC}_{i-1}^{\lll 0} \oplus \mathsf{RC}_{i-1}^{\lll 80} \oplus \mathsf{RC}_{i}^{\lll 0} \oplus \mathsf{RC}_{i}^{\lll 1} \oplus \mathsf{RC}_{i}^{\lll 81}$
$mask_b^{(i-2)}$	$\mathsf{RC}_{i-1}^{\lll 1} \oplus \mathsf{RC}_{i-1}^{\lll 80} \oplus \mathsf{RC}_{i-1}^{\lll 81} \oplus \mathsf{RC}_{i}^{\lll 2} \oplus \mathsf{RC}_{i}^{\lll 80} \oplus \mathsf{RC}_{i}^{\lll 81} \oplus \mathsf{RC}_{i}^{\lll 82}$
$mask_c^{(i-2)}$	$\mathsf{RC}_{i-2}^{\lll 0} \oplus \mathsf{RC}_{i-1}^{\lll 0} \oplus \mathsf{RC}_{i-1}^{\lll 1} \oplus \mathsf{RC}_{i-1}^{\lll 81} \oplus \mathsf{RC}_{i}^{\lll 1} \oplus \mathsf{RC}_{i}^{\lll 2} \oplus \mathsf{RC}_{i}^{\lll 82}$
$mask_a^{(i-1)}$	$\mathsf{RC}_{i}^{\lll 0} \oplus \mathsf{RC}_{i}^{\lll 80}$
$mask_b^{(i-1)}$	$\mathsf{RC}_{i}^{\lll 1} \oplus \mathsf{RC}_{i}^{\lll 80} \oplus \mathsf{RC}_{i}^{\lll 81}$
$mask_c^{(i-1)}$	$\mathsf{RC}_{i-1}^{\lll 0} \oplus \mathsf{RC}_{i}^{\lll 0} \oplus \mathsf{RC}_{i}^{\lll 1} \oplus \mathsf{RC}_{i}^{\lll 81}$
$mask_a^{(i)}$	0
$mask_b^{(i)}$	0
$mask_c^{(i)}$	$\mathsf{RC}_{i}^{\lll 0}$
$mask_a^{(i+1)}$	0
$mask_b^{(i+1)}$	0
$mask_c^{(i+1)}$	0
$mask_a^{(i+2)}$	$\mathsf{RC}_{i+1}^{\lll 0}$
$mask_b^{(i+2)}$	$\mathsf{RC}_{i+1}^{\lll 80}$
$mask_c^{(i+2)}$	$\mathsf{RC}_{i+1}^{\lll 80}$
$mask_a^{(i+3)}$	$\mathsf{RC}_{i+1}^{\lll 0} \oplus \mathsf{RC}_{i+2}^{\lll 0}$
$mask_b^{(i+3)}$	$\mathsf{RC}_{i+1}^{\lll 1} \oplus \mathsf{RC}_{i+1}^{\lll 80} \oplus \mathsf{RC}_{i+1}^{\lll 81} \oplus \mathsf{RC}_{i+1}^{\lll 160} \oplus \mathsf{RC}_{i+2}^{\lll 80}$
$mask_c^{(i+3)}$	$\mathsf{RC}_{i+1}^{\lll 0} \oplus \mathsf{RC}_{i+1}^{\lll 81} \oplus \mathsf{RC}_{i+1}^{\lll 160} \oplus \mathsf{RC}_{i+2}^{\lll 80}$
$mask_a^{(i+4)}$	$\mathsf{RC}_{i+1}^{\lll 1} \oplus \mathsf{RC}_{i+1}^{\lll 80} \oplus \mathsf{RC}_{i+2}^{\lll 0}$
$mask_b^{(i+4)}$	$\mathsf{RC}_{i+1}^{\lll 160} \oplus \mathsf{RC}_{i+1}^{\lll 161} \oplus \mathsf{RC}_{i+1}^{\lll 240} \oplus \mathsf{RC}_{i+2}^{\lll 1} \oplus \mathsf{RC}_{i+2}^{\lll 80} \oplus \mathsf{RC}_{i+2}^{\lll 81} \oplus \mathsf{RC}_{i+2}^{\lll 160} \oplus \mathsf{RC}_{i+3}^{\lll 0} \oplus \mathsf{RC}_{i+3}^{\lll 80}$
$mask_c^{(i+4)}$	$\mathsf{RC}_{i+1}^{\lll 0} \oplus \mathsf{RC}_{i+1}^{\lll 80} \oplus \mathsf{RC}_{i+1}^{\lll 81} \oplus \mathsf{RC}_{i+1}^{\lll 161} \oplus \mathsf{RC}_{i+1}^{\lll 240} \oplus \mathsf{RC}_{i+2}^{\lll 0} \oplus \mathsf{RC}_{i+2}^{\lll 81} \oplus \mathsf{RC}_{i+2}^{\lll 160} \oplus \mathsf{RC}_{i+3}^{\lll 80}$
$mask_a^{(i+5)}$	$\mathsf{RC}_{i+1}^{\lll 0} \oplus \mathsf{RC}_{i+1}^{\lll 1} \oplus \mathsf{RC}_{i+1}^{\lll 81} \oplus \mathsf{RC}_{i+1}^{\lll 160} \oplus \mathsf{RC}_{i+2}^{\lll 1} \oplus \mathsf{RC}_{i+2}^{\lll 80} \oplus \mathsf{RC}_{i+3}^{\lll 0} \oplus \mathsf{RC}_{i+4}^{\lll 0}$
$mask_b^{(i+5)}$	$\mathsf{RC}_{i+1}^{\lll 2} \oplus \mathsf{RC}_{i+1}^{\lll 80} \oplus \mathsf{RC}_{i+1}^{\lll 81} \oplus \mathsf{RC}_{i+1}^{\lll 82} \oplus \mathsf{RC}_{i+1}^{\lll 161} \oplus \mathsf{RC}_{i+1}^{\lll 240} \oplus \mathsf{RC}_{i+1}^{\lll 320} \oplus \mathsf{RC}_{i+2}^{\lll 160} \oplus \mathsf{RC}_{i+2}^{\lll 161} \oplus \mathsf{RC}_{i+2}^{\lll 240} \oplus \mathsf{RC}_{i+3}^{\lll 0} \oplus \mathsf{RC}_{i+3}^{\lll 80} \oplus \mathsf{RC}_{i+3}^{\lll 160} \oplus \mathsf{RC}_{i+4}^{\lll 80}$
$mask_c^{(i+5)}$	$\mathsf{RC}_{i+1}^{\lll 1} \oplus \mathsf{RC}_{i+1}^{\lll 82} \oplus \mathsf{RC}_{i+1}^{\lll 160} \oplus \mathsf{RC}_{i+1}^{\lll 241} \oplus \mathsf{RC}_{i+1}^{\lll 320} \oplus \mathsf{RC}_{i+2}^{\lll 0} \oplus \mathsf{RC}_{i+2}^{\lll 80} \oplus \mathsf{RC}_{i+2}^{\lll 81} \oplus \mathsf{RC}_{i+2}^{\lll 161} \oplus \mathsf{RC}_{i+2}^{\lll 240} \oplus \mathsf{RC}_{i+3}^{\lll 160} \oplus \mathsf{RC}_{i+4}^{\lll 80}$

Further Discussion for the Online Phase. According to Theorem 1, the lower the hamming weight in the rotational pair associated with the influence of the round constants, the higher is the rotation probability of the AND operation in the presence of round constants; thus, if we mitigate the influence of the round constants as much as possible, we can perform the online phase in the proposed attack procedure with a high probability. To mitigate the influence of the round constants in the online phase, we deliberate over the following three questions:

Q1. Should we select the pattern $(X, \overleftarrow{X})$ or $(X, \overrightarrow{X})$ as a rotational pair?
Q2. What value should we select as a rotational amount r?
Q3. How should we decide the target rounds?

To answer these questions, we analyze the round constants of Friet-PC by using the following four examples:

Example 3. We consider the case where exactly one bit is 1 in the round constants of Friet-PC, such as rc_9, rc_{10}, rc_{11}. In this example, we use rc_9 for the sake of simplicity. Then, the hamming weight of $[\mathsf{rc}_9 \oplus \overleftarrow{\mathsf{rc}_9}]$ can be minimized regardless of the selection of the rotational pair and rotational amount, i.e., $hw[\mathsf{rc}_9 \oplus \overleftarrow{\mathsf{rc}_9}] = 2$.

Example 4. We consider the case where two or more bits are 1 in the round constants of Friet-PC and all of the bit strings 1 are continuous in hexadecimal notation, such as rc_0, rc_1, rc_6. In this example, we use rc_0 for the sake of simplicity. Then, the hamming weight of $[\mathsf{rc}_0 \oplus \overleftarrow{\mathsf{rc}_0}]$ can be minimized when the rotational amount is selected as $r = 4$, regardless of the selection of the rotational pair, i.e., $hw[\mathsf{rc}_0 \oplus \overleftarrow{\mathsf{rc}_0}] = 2$. If the rotational amount is selected as $r = 1$, the hamming weight of $[\mathsf{rc}_0 \oplus \overleftarrow{\mathsf{rc}_0}]$ can be maximized, e.g., $hw[\mathsf{rc}_0 \oplus \overleftarrow{\mathsf{rc}_0}] = 8$.

Example 5. We consider the case where two bits are 1 in the round constants of Friet-PC and the bit strings 1 are not continuous in hexadecimal notation, such as rc_3, rc_4, rc_8. In this example, we use rc_3 and rc_8 for the sake of simplicity. In one case, the hamming weight of $[\mathsf{rc}_3 \oplus \overleftarrow{\mathsf{rc}_3}]$ can be minimized when the rotational amount is selected as $r = 8$, regardless of the selection of the rotational pair, i.e., $hw[\mathsf{rc}_3 \oplus \overleftarrow{\mathsf{rc}_3}] = 2$. In another case, the hamming weight of $[\mathsf{rc}_8 \oplus \overleftarrow{\mathsf{rc}_8}]$ can be minimized when the rotational amount is selected as $r = 12$, regardless of the selection of the rotational pair, i.e., $hw[\mathsf{rc}_8 \oplus \overleftarrow{\mathsf{rc}_8}] = 2$. Therefore, the distance between 2-bit strings 1 is the optimum rotational amount.

Example 6. We consider the case where three or more bits are 1 in the round constants of Friet-PC and the bit strings 1 are not continuous in hexadecimal notation, such as rc_2, rc_5, rc_{17}. In this example, we use rc_2 for the sake of simplicity. Then, the hamming weight of $[\mathsf{rc}_2 \oplus \overleftarrow{\mathsf{rc}_2}]$ can be minimized when the rotational amount is selected as $r = 4$, regardless of the selection of the rotational pair, i.e., $hw[\mathsf{rc}_2 \oplus \overleftarrow{\mathsf{rc}_2}] = 4$.

These examples show that to mitigate the influence of the round constant, we need to change the rotational amount according to the value of the round constant though we can freely select the rotational pair; however, it is impossible to change the rotational amount while performing a rotational attack. Hence, we need to decide the target round that can mitigate the influence of the round constants without changing the rotational amount. Consequently, we choose the 9th to 16th round of Friet-PC as the target rounds in order to efficiently perform the online phase in the proposed attack on the 8-round Friet-PC. As discussed in Examples 3 and 4, for the round constants in the target rounds, the hamming weight can be minimized by selecting the rotational amount as $r = 4$. In addition, we select the pattern $(X, \overleftarrow{X})$ as the rotational pair.

Table 6. Minimum weights for the AND operation in the target round of FRIET-PC.

Round i	9	10	11	12	13	14	15	16	sum
Weight	28	14	2	0	4	18	36	0	102

Complexity Estimation. As discussed in Section 3.2, to perform a rotational attack on FRIET-PC properly, we need to evaluate the rotational probability of the AND operation in the presence of round constants. When focusing on the round function of FRIET-PC, only the output limb a is influenced by the AND operation. Further, according to Algorithm 1, the AND operation is executed in the final step of the round function of FRIET-PC, and the output limbs (b, c) in each round become the input of its AND operation; thus, this situation implies that the output mask values $(mask_b^{(r+1)}, mask_c^{(r+1)})$ for the r-th round output limbs (b, c) influence a rotational probability of the AND operation in each round.

Based on Theorem 1, we estimate a rotational probability of the AND operation in the round function of FRIET-PC by calculating the hamming weight from $(mask_b^{(r+1)}, mask_c^{(r+1)})$ for $r \in \{i-3, i-2, i-1, i, i+1, i+2, i+3\}$ and $i = 12$. Table 6 lists the minimum hamming weights for the AND operation in the target round of FRIET-PC. As discussed earlier, we can estimate the minimum hamming weights for each mask values, such as $hw[\mathsf{RC}_i^{\lll_0}] = 2$, by selecting the rotational amount as $r = 4$. To confirm the accuracy of our estimation, we have conducted an experiment to compute the rotational probability of the 10th to 14th round of FRIET-PC; then, we have confirmed that the rotational probability of the target round can be approximated to 2^{-38}. Herein, we explain that the minimum hamming weight in the 16th round of FRIET-PC is 0. This is because the output limbs (b, c) in each round are not influenced by the AND operation; thus, when a complete rotational pair holds for all input limbs (a, b, c) in each round, a rotational distinguisher can be performed with a probability of one by masking properly the output limbs (b, c) with the mask values listed in Table 5 (experimentally verified over 2^{32} trials).

To summarize our results, we choose the 9th to 16th round of FRIET-PC as the target rounds, and have demonstrated a rotational distinguisher for the 8-round FRIET-PC with a time complexity of 2^{102}. However, we cannot demonstrate a rotational distinguisher for 9 or more rounds of FRIET-PC because it provides a 128-bit security level.

4 Bit-wise Differential Distinguisher

In this section, we investigate the security of FRIET-PC against a bit-wise differential attack, which has been mainly applied to ARX ciphers, such as stream ciphers Salsa and ChaCha [2, 6, 24]. Specifically, we focus on single- and dual-bit differential attacks, reported by Choudhuri and Maitra [6], and demonstrate a practical bit-wise differential distinguisher for the 9-round FRIET-PC with a time complexity of $2^{20.044}$.

4.1 Single- and Dual-bit Differential Attacks

Let $x_i^{(r)}$ be the i-th bit of the r-round limb x for $0 \leq i \leq 127$; let $x_i'^{(r)}$ be an associated bit with the difference $\Delta x_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$. The input difference $\Delta x_i^{(0)}$ for $r = 0$ and the output difference $\Delta x_i^{(r)}$ for $r > 0$ are referred to as $\mathcal{ID}$ and $\mathcal{OD}$, respectively. We note that $x_0^{(r)}$ and $x_{127}^{(r)}$ are the least significant bit (LSB) and most significant bit (MSB), respectively. For all possible choices of input limbs, single- and dual-bit differential probabilities are defined by

$$\Pr\big(\Delta x_j^{(r)} = 1 \mid \Delta x_i^{(0)} = 1\big) = \frac{1}{2}(1 + \epsilon_d), \tag{15}$$

$$\Pr\big(\Delta x_{j_0}^{(r)} \oplus \Delta x_{j_1}^{(r)} = 1 \mid \Delta x_i^{(0)} = 1\big) = \frac{1}{2}(1 + \epsilon_d), \tag{16}$$

where ϵ_d denotes the bias of the $\mathcal{OD}$.

To distinguish the r-round limb $x^{(r)}$ computed by the reduced-round Friet-PC from true random number sequences, we use the following theorem proved by Mantin and Shamir [21].

Theorem 2 ([21, Theorem 2]). *Let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions, and suppose that the event e occurs in $\mathcal{X}$ with a probability p and $\mathcal{Y}$ with a probability $p\cdot(1+q)$. Then, for small p and q, $\mathcal{O}(\frac{1}{p\cdot q^2})$ samples suffice to distinguish $\mathcal{X}$ from $\mathcal{Y}$ with a constant probability of success.*

Let $\mathcal{X}$ be a distribution of $\mathcal{OD}$ of true random number sequences, and let $\mathcal{Y}$ be a distribution of $\mathcal{OD}$ of the reduced-round Friet-PC. Based on single-bit and dual-bit differential probabilities, the number of samples to distinguish $\mathcal{X}$ and $\mathcal{Y}$ is $\mathcal{O}(\frac{2}{\epsilon_d^2})$ since p and q are equal to $\frac{1}{2}$ and ϵ_d, respectively.

4.2 Experimental Results

To find bit-wise differential biases of the reduced-round Friet-PC, we have conducted experiments with 2^{28} randomly chosen samples. Our experimental environment is as follows: five Linux machines with 40-core Intel(R) Xeon(R) CPU E5-2660 v3 (2.60 GHz), 128.0 GB of main memory, a gcc 7.2.0 compiler, and the C programming language.

Tables 7-9 list the single- and dual-bit differential biases for the 9-, 10-, and 11-round Friet-PC. As shown in Table 7, we obtain the best bit-wise differential bias for the 9-round Friet-PC, such that $\mathcal{ID}$ is $\Delta b_{40}^{(0)}$, $\mathcal{OD}$ is $\Delta a_{121}^{(9)} \oplus \Delta c_{54}^{(9)}$, and ϵ_d is approximately $2^{-9.360}$. To obtain a more precise differential bias in this $\mathcal{ID}$-$\mathcal{OD}$ pair, we have conducted an additional experiment with 2^{36} randomly chosen samples. Thus, we obtain a more precise differential bias for the 9-round Friet-PC, such that ϵ_d is approximately $2^{-9.522}$. According to Theorem 2, $2^{20.044}$ samples are sufficient for distinguishing the 9-round Friet-PC from a true random number generator with a constant probability of success. For the 9-round Friet-PC, the best dual-bit differential bias, i.e., $\epsilon_d = 2^{-9.522}$, provides a practical bit-wise differential distinguisher when $\mathcal{ID}$ is $\Delta b_{40}^{(0)}$ and $\mathcal{OD}$

Table 7. Single- and dual-bit differential biases ($\log_2$) for the 9-round Friet-PC.

Single-bit			Dual-bit		
$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d
$\Delta b^{(0)}_{113}$	$\Delta a^{(9)}_{66}$	-9.390	$\Delta b^{(0)}_{40}$	$\Delta a^{(9)}_{121} \oplus \Delta c^{(9)}_{54}$	-9.360
$\Delta b^{(0)}_{109}$	$\Delta a^{(9)}_{62}$	-9.395	$\Delta b^{(0)}_{105}$	$\Delta a^{(9)}_{58} \oplus \Delta c^{(9)}_{119}$	-9.378
$\Delta b^{(0)}_{63}$	$\Delta a^{(9)}_{16}$	-9.415	$\Delta b^{(0)}_{118}$	$\Delta a^{(9)}_{71} \oplus \Delta b^{(9)}_{35}$	-9.395
$\Delta b^{(0)}_{125}$	$\Delta a^{(9)}_{78}$	-9.417	$\Delta b^{(0)}_{14}$	$\Delta a^{(9)}_{95} \oplus \Delta b^{(9)}_{59}$	-9.413
$\Delta b^{(0)}_{23}$	$\Delta a^{(9)}_{104}$	-9.421	$\Delta b^{(0)}_{38}$	$\Delta a^{(9)}_{119} \oplus \Delta c^{(9)}_{52}$	-9.414
$\Delta b^{(0)}_{78}$	$\Delta a^{(9)}_{31}$	-9.430	$\Delta b^{(0)}_{92}$	$\Delta a^{(9)}_{45} \oplus \Delta c^{(9)}_{106}$	-9.416
$\Delta b^{(0)}_{58}$	$\Delta a^{(9)}_{11}$	-9.434	$\Delta b^{(0)}_{28}$	$\Delta a^{(9)}_{109} \oplus \Delta c^{(9)}_{42}$	-9.420
$\Delta b^{(0)}_{98}$	$\Delta a^{(9)}_{51}$	-9.436	$\Delta b^{(0)}_{55}$	$\Delta a^{(9)}_{8} \oplus \Delta b^{(9)}_{100}$	-9.423

Table 8. Single- and dual-bit differential biases ($\log_2$) for the 10-round Friet-PC.

Single-bit			Dual-bit		
$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d
$\Delta c^{(0)}_{111}$	$\Delta a^{(10)}_{83}$	-11.863	$\Delta a^{(0)}_{118}$	$\Delta b^{(10)}_{122} \oplus \Delta c^{(10)}_{45}$	-11.501
$\Delta c^{(0)}_{10}$	$\Delta c^{(10)}_{117}$	-11.872	$\Delta c^{(0)}_{57}$	$\Delta b^{(10)}_{4} \oplus \Delta b^{(10)}_{117}$	-11.509
$\Delta c^{(0)}_{125}$	$\Delta a^{(10)}_{124}$	-11.897	$\Delta c^{(0)}_{102}$	$\Delta b^{(10)}_{27} \oplus \Delta b^{(10)}_{120}$	-11.588
$\Delta c^{(0)}_{66}$	$\Delta a^{(10)}_{104}$	-11.921	$\Delta c^{(0)}_{119}$	$\Delta b^{(10)}_{109} \oplus \Delta c^{(10)}_{79}$	-11.609
$\Delta a^{(0)}_{21}$	$\Delta c^{(10)}_{120}$	-11.931	$\Delta c^{(0)}_{99}$	$\Delta a^{(10)}_{31} \oplus \Delta c^{(10)}_{16}$	-11.617
$\Delta b^{(0)}_{52}$	$\Delta b^{(10)}_{58}$	-11.953	$\Delta b^{(0)}_{103}$	$\Delta a^{(10)}_{58} \oplus \Delta b^{(10)}_{47}$	-11.632
$\Delta c^{(0)}_{16}$	$\Delta b^{(10)}_{72}$	-11.954	$\Delta b^{(0)}_{83}$	$\Delta a^{(10)}_{6} \oplus \Delta b^{(10)}_{41}$	-11.640
$\Delta c^{(0)}_{122}$	$\Delta a^{(10)}_{13}$	-11.960	$\Delta b^{(0)}_{113}$	$\Delta a^{(10)}_{8} \oplus \Delta c^{(10)}_{103}$	-11.649

is $\Delta a^{(9)}_{121} \oplus \Delta c^{(9)}_{54}$. Similarly, as shown in Tables 8 and 9, we obtain the best bit-wise differential biases for the 10- and 11-round Friet-PC, such that ϵ_d are approximately $2^{-11.501}$ and $2^{-11.596}$, respectively. These experimental results may indicate insufficient accuracy because the best differential biases for the 10- and 11-round Friet-PC are approximately equal. To obtain a more precise differential bias for the 10-round Friet-PC, we have conducted an additional experiment with 2^{38} randomly chosen samples when $\mathcal{ID}$ is $\Delta a^{(0)}_{118}$ and $\mathcal{OD}$ is $\Delta b^{(10)}_{122} \oplus \Delta c^{(10)}_{45}$. This is the best $\mathcal{ID}$-$\mathcal{OD}$ pair for the 10-round Friet-PC. Consequently, we obtain the more precise differential bias for the 10-round Friet-PC, such that ϵ_d is approximately $2^{-18.634}$; thus, at least $2^{38.268}$ samples are sufficient for distinguishing the 10-round Friet-PC from a true random number generator with a constant probability of success. In summary, our experiments have revealed that the practical bit-wise differential distinguisher for Friet-PC performs properly up to 9 rounds (out of 24 rounds in the original version).

Table 9. Single- and dual-bit differential biases ($\log_2$) for the 11-round Friet-PC.

Single-bit			Dual-bit		
$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d
$\Delta a_{100}^{(0)}$	$\Delta a_{74}^{(11)}$	-11.772	$\Delta a_{48}^{(0)}$	$\Delta a_{19}^{(11)} \oplus \Delta c_{103}^{(11)}$	-11.596
$\Delta b_{57}^{(0)}$	$\Delta a_{52}^{(11)}$	-11.844	$\Delta c_{111}^{(0)}$	$\Delta b_{57}^{(11)} \oplus \Delta c_{19}^{(11)}$	-11.604
$\Delta c_{14}^{(0)}$	$\Delta a_{73}^{(11)}$	-11.847	$\Delta c_{25}^{(0)}$	$\Delta b_{5}^{(11)} \oplus \Delta b_{90}^{(11)}$	-11.616
$\Delta c_{21}^{(0)}$	$\Delta a_{3}^{(11)}$	-11.873	$\Delta c_{89}^{(0)}$	$\Delta a_{90}^{(11)} \oplus \Delta b_{73}^{(11)}$	-11.635
$\Delta a_{20}^{(0)}$	$\Delta a_{61}^{(11)}$	-11.885	$\Delta b_{31}^{(0)}$	$\Delta c_{17}^{(11)} \oplus \Delta c_{59}^{(11)}$	-11.647
$\Delta b_{84}^{(0)}$	$\Delta b_{104}^{(11)}$	-11.886	$\Delta b_{67}^{(0)}$	$\Delta a_{68}^{(11)} \oplus \Delta c_{60}^{(11)}$	-11.655
$\Delta a_{45}^{(0)}$	$\Delta b_{37}^{(11)}$	-11.913	$\Delta b_{88}^{(0)}$	$\Delta a_{20}^{(11)} \oplus \Delta c_{11}^{(11)}$	-11.679
$\Delta b_{56}^{(0)}$	$\Delta b_{108}^{(11)}$	-11.913	$\Delta c_{74}^{(0)}$	$\Delta a_{60}^{(11)} \oplus \Delta b_{5}^{(11)}$	-11.681

5 MILP-aided Zero-sum Distinguisher

5.1 Zero-sum Distinguisher and Division Property

The zero-sum distinguisher [3] a widely-utilized tool to evaluate the security of a public permutation, though it has never influenced the security of the corresponding hash or encryption schemes as far as we know. A critical reason exists in the attackers' capacity to control the whole internal state, which is impossible in the schemes adopting the sponge structure. However, it is still interesting if one could identify a non-trivial zero-sum distinguisher with better time complexity than those obtained with trivial algebraic degree evaluations.

The bit-based division property [27] is a powerful technique to compute the increase of algebraic degrees for a bit-oriented public permutation, especially when combined with the automatic search method [28]. However, the usage of division property has not been discussed in the proposal of Friet [26] and we believe this is essential if non-trivial increase of algebraic degrees could be identified. Consequently, in the following part, we briefly introduce bit-based division property [27] and then report our findings.

First, define the following functions before defining the division property.

Definition 1 (Bit-Product Function). *For any $u \in \mathbb{F}_2^n$, let $\pi_u(x)$ be a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. For any $x \in \mathbb{F}_2^n$, define $\pi_u(x)$ as follows:*

$$\pi_u(x) = \prod_{i=0}^{n-1} x[i]^{u[i]}$$

Let $\boldsymbol{\pi_u}$ be a function from $(\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_{m-1}}\ )$ to $\mathbb{F}^2$ for all $\boldsymbol{u} \in \mathbb{F}_2^n$. For any $\boldsymbol{u} = (u_0, u_1, \cdots, u_{m-1}), \boldsymbol{x} = (x_0, x_1, \cdots, x_{m-1})$, define $\boldsymbol{\pi_u}(\boldsymbol{x})$ as follows.

$$\boldsymbol{\pi_u}(\boldsymbol{x}) = \prod_{i=0}^{m-1} \pi_{u_i}(x_i)$$

Then, the bit-based division property [27] can be defined as follows:

Definition 2 (Bit-Based Division Property). *Let $\mathbb{X}$ be a multiset whose elements takes a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, where $\mathbb{K}$ denotes a set of n-dimensional vectors whose i-th element takes 0 or 1, it fulfills the following conditions:*

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \begin{cases} \mathit{unknown} \text{ if there exist } \boldsymbol{k}\in\mathbb{K} \text{ s.t. } wt(\boldsymbol{u}) \succeq \boldsymbol{k}, \\ 0 \qquad\qquad \mathit{otherwise}. \end{cases}$$

$wt(\boldsymbol{u})$ is the hamming weight of $\boldsymbol{u}$. If there $\boldsymbol{k}\in\mathbb{K}$ and $\boldsymbol{k}'\in\mathbb{K}$ satisfying $\boldsymbol{k}\succeq\boldsymbol{k}'$ in the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, $\boldsymbol{k}$ can be removed from $\mathbb{K}$ because it is redundant. When we utilize MILP method to evaluate the division property propagation, we need to focus on the elements of $\mathbb{K}$. Xiang et al. proposed new notations [28] called division trail to illustrate division property propagation, which can be defined as follows:

Definition 3 (Division Trail). *Let f_r denote the round function of an iterated block cipher. Assume the input multiset to the block cipher has initial division property $\mathcal{D}_{\boldsymbol{k}}^{n,m}$, and denote the division property after i-round propagation through f_r by $\mathcal{D}_{\mathbb{K}_i}^{n,m}$. Thus, we have the following chain of division property propagations:*

$$\{\boldsymbol{k}\} \overset{def}{=} \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} \cdots$$

Moreover, for any vector $\boldsymbol{k}_i^*$ in $\mathbb{K}_i (i \geq 1)$, there must exist an vector $\boldsymbol{k}_{i-1}^*$ in $\mathbb{K}_{i-1}$ such that $\boldsymbol{k}_{i-1}^*$ can propagate to $\boldsymbol{k}_i^*$ by division property propagation rules. Furthermore, for $(\boldsymbol{k}_0, \boldsymbol{k}_1, \cdots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$, if $\boldsymbol{k}_{i-1}$ can propagate to $\boldsymbol{k}_i$ for all $i \in \{1, 2, \cdots, r\}$, we call $(\boldsymbol{k}_0, \boldsymbol{k}_1, \cdots, \boldsymbol{k}_r)$ an r-round division trail.

Proposition 1. *Denote the division property of input mulitset to an iterated block cipher by $\mathcal{D}_{\boldsymbol{k}}^{n,m}$, let f_r be the round function. Denote*

$$\{\boldsymbol{k}\} \overset{def}{=} \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} \cdots \xrightarrow{f_r} \mathbb{K}_r$$

the r-round division property propagation. Thus, the set of the last vectors of all r-round division trails which start with $\boldsymbol{k}$ is equal to $\mathbb{K}_r$.

In general, we need to show that the Hamming weight of any vector of $\mathbb{K}_r$ derived from the division property $\mathcal{D}_{\mathbb{K}_0}$ of input multiset is not less than or equal to 1, and then we need to prove that the division trail where $\mathbb{K}_r$ is *unknown* does not exist.

5.2 MILP Modeling

In this subsection, we describe the MILP-based methods to search for the integral distinguishers [7] and explain how to express the division property propagation through the basic operations of Friet-PC based on the method proposed by Xiang et al. [28].

When evaluating the propagation of division property, it is necessary to consider the basic operations of a block cipher such as COPY and XOR. In the following, we will introduce the bit-based division property propagation through these basic operations and how to express the division property propagation through these operations as linear inequalities.

Modeling COPY. COPY operation is the basic operation used in Feistel ciphers. A portion of the input copied into two equal parts, one of which is fed to the round function. Denote F an function taking $x \in \mathbb{F}_2$ as input and $(y_0, y_1) = (x, x)$ as output. If the input multiset $\mathbb{X}$ has division property $\mathcal{D}_k^n$, the output multiset $\mathbb{Y}$ will have division property $\mathcal{D}_{\mathbb{K}'}^{n,n}$, where

$$\mathbb{K}' \leftarrow (k - i, i) \text{ for } 0 \leq i \leq k.$$

Since we consider the bit-based division property, we only need to consider the division property propagation where $k = 1$. Thus, the division trails are $(0) \overset{copy}{\rightarrow} (0, 0)$, $(1) \overset{copy}{\rightarrow} (0, 1)$ and $(1) \overset{copy}{\rightarrow} (1, 0)$. Let $(a) \overset{copy}{\rightarrow} (b_0, b_1)$ be the division trails through the COPY operation, the following inequalities are sufficient to describe the division property propagation of COPY [28].

$$\begin{cases} a - b_0 - b_1 = 0 \\ a, b_0, b_1 \text{ are binaries} \end{cases}$$

Modeling XOR. Denote F an function taking $(x_0, x_1) \in \mathbb{F}_2 \times \mathbb{F}_2$ as input and $y = x_0 \oplus x_1$ as output. If the input multiset $\mathbb{X}$ has division property $\mathcal{D}_{\mathbb{K}}^{n,n}$, the output multiset $\mathbb{Y}$ will have division property $\mathcal{D}_{k'}^n$, where

$$k' = \min_{(k_0, k_1) \in \mathbb{K}} \{k_0 + k_1\}$$

We only need to consider the division property propagation where $k = 1$. Thus, the division trails are $(0, 0) \overset{XOR}{\rightarrow} (0)$, $(0, 1) \overset{XOR}{\rightarrow} (1)$ and $(1, 0) \overset{XOR}{\rightarrow} (1)$. Let $(a_0, a_1) \overset{XOR}{\rightarrow} (b)$ be the division trails through the XOR operation, the following inequalities are sufficient to describe the division property propagation of XOR [28]:

$$\begin{cases} a_0 + a_1 - b = 0 \\ a_0, a_1, b \text{ are binaries} \end{cases}$$

Modeling AND. Denote F an function taking $(x_0, x_1) \in \mathbb{F}_2 \times \mathbb{F}_2$ as input and $y = x_0 \wedge x_1$ as output. If the input multiset $\mathbb{X}$ has division property $\mathcal{D}_{\mathbb{K}}^{n,n}$, the output multiset $\mathbb{Y}$ will have division property $\mathcal{D}_{k'}^n$, where

$$\mathbb{K}' \leftarrow \max_{(k_0, k_1) \in \mathbb{K}} \{k_0, k_1\}$$

We only need to consider the division property propagation where $k = 1$. Thus, the division trails are $(0, 0) \overset{AND}{\rightarrow} (0)$, $(0, 1) \overset{AND}{\rightarrow} (1)$, $(1, 0) \overset{AND}{\rightarrow} (1)$, $(1, 1) \overset{AND}{\rightarrow}$

Table 10. Integral distinguishers

Complexity	Rounds	Integral distinguisher
2^{31}	7	$(\mathcal{C}^{128}, \mathcal{C}^{97}\mathcal{A}^{31}, \mathcal{C}^{128}) \rightarrow (\mathcal{U}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$
2^{63}	8	$(\mathcal{C}^{128}, \mathcal{C}^{65}\mathcal{A}^{63}, \mathcal{C}^{128}) \rightarrow (\mathcal{U}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$
2^{127}	9	$(\mathcal{C}^{128}, \mathcal{C}\mathcal{A}^{127}, \mathcal{C}^{128}) \rightarrow (\mathcal{U}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$
2^{255}	9	$(\mathcal{C}^{128}, \mathcal{A}^{128}, \mathcal{C}\mathcal{A}^{127}) \rightarrow (\mathcal{U}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$
2^{383}	15	$(\mathcal{C}\mathcal{A}^{127}, \mathcal{A}^{128}, \mathcal{A}^{128}) \rightarrow (\mathcal{B}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$

(1). Let $(a_0, a_1) \overset{AND}{\rightarrow} (b)$ be the division trails through the AND operation, the following inequalities are sufficient to describe the division property propagation of AND [28]:

$$\begin{cases} b - a_0 \geq 0 \\ b - a_1 \geq 0 \\ b - a_0 - a_1 \leq 0 \\ a_0, a_1, b \ \text{ are binaries} \end{cases}$$

The Initial Division Property. Since we search for integral distinguishers based on bit-based division property, it is necessary to set the input division property to ALL ($\mathcal{A}$) or CONSTANT ($\mathcal{C}$) for each bit independently. Assuming we have 2^s plaintext, we can set s bits in the initial division property as ALL ($\mathcal{A}$).

Stopping Rule. Let $(a^0_{n-1}, \cdots, a^0_0) \rightarrow \cdots \rightarrow (a^r_{n-1}, \cdots, a^r_0)$ be a r-round division trail. If the trail where the output division property with only i-th bit $(0 \leq i < n)$ being 1 and the rest being 0 for a given initial division property does not exist, the i-th bit holds the BALANCE ($\mathcal{B}$) property. We can check whether i-th bit holds BALANCE ($\mathcal{B}$) or UNKNOWN ($\mathcal{U}$) by checking if such a trail exists. This can be easily evaluated with MILP [28]. Specifically, if the model is infeasible for the given constraints, there is no such trail, and vice versa.

5.3 Our Search

We modeled the operations of the Friet-PC round as the MILP constraints and optimized the models using the MILP solver. All the models are solved with the Gurobi solver [11]. All the searches are performed on a machine equipped with an Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz with HyperThreading enabled.

From Fig. 1, it is clear that the input of limb b will not pass through the AND operation, which is the only non-linear transformation part in the round function. Therefore, for zero-sum distinguisher with a low time complexity, it is always better to choose as many active bits from limb b as possible. The obtained integral distinguishers are shown in Table 10.

5.4 Zero-sum Distinguishers

The above integral distinguisher can be converted into zero-sum distinguishers with a start-from-the-middle method as in [3]. Specifically, we view an internal state in a middle round as input and search for integral distinguishers in both backward and forward directions. As a result, the following four zero-sum distinguishers can be constructed:

– 30-round zero-sum distinguisher with 2^{383} time and data complexity.

$$(\mathcal{B}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128}) \overset{15-round}{\leftarrow} (\mathcal{C}\mathcal{A}^{127}, \mathcal{A}^{128}, \mathcal{A}^{128}) \overset{15-round}{\rightarrow} (\mathcal{B}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$$

– 17-round zero-sum distinguisher with 2^{127} time and data complexity.

$$(\mathcal{B}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128}) \overset{8-round}{\leftarrow} (\mathcal{C}^{128}, \mathcal{C}\mathcal{A}^{127}, \mathcal{C}^{128}) \overset{9-round}{\rightarrow} (\mathcal{U}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$$

– 15-round zero-sum distinguisher with 2^{63} time and data complexity.

$$(\mathcal{B}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128}) \overset{7-round}{\leftarrow} (\mathcal{C}^{128}, \mathcal{C}^{65}\mathcal{A}^{63}, \mathcal{C}^{128}) \overset{8-round}{\rightarrow} (\mathcal{U}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$$

– 13-round zero-sum distinguisher with 2^{31} time and data complexity.

$$(\mathcal{B}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128}) \overset{6-round}{\leftarrow} (\mathcal{C}^{128}, \mathcal{C}^{97}\mathcal{A}^{31}, \mathcal{C}^{128}) \overset{7-round}{\rightarrow} (\mathcal{U}^{128}, \mathcal{B}^{128}, \mathcal{B}^{128})$$

In summary, a practical 13-round zero-sum distinguisher and a theoretical 17-round zero-sum distinguisher with time complexity below 2^{128} are obtained. However, the full-round zero-sum distinguisher requires half of the total input space, i.e., it requires 2^{383} time and data.

Remark. It is in general difficult to compare distinguishers on a public permutation if the attacker has a control over the full internal state, as this is always impossible in schemes constructed with a public permutation and the sponge structure. Notice that the distinguishing attacks reported in [19] also require the capability to control the whole internal state of Friet-PC.

6 Conclusion

In this study, we evaluated the security of the Friet-PC permutation against bit-wise cryptanalysis including rotational, bit-wise differential, and integral attacks. First, we provided a generic procedure for a rotational attack on AND-RX ciphers with round constants and applied it to the Friet-PC permutation. Subsequently, we demonstrated an 8-round rotational distinguisher with a time complexity of 2^{102}. Second, we explored single- and dual-bit differential biases of the reduced-round Friet-PC and extended one of them to a 9-round bit-wise differential distinguisher with a time complexity of $2^{20.044}$. Finally, we found 7-, 8-, 9-, and 15-round integral characteristics and extended these characteristics to 13-, 15-, 17-, and 30-round zero-sum distinguishers with time complexities of

2^{31}, 2^{63}, 2^{127}, and 2^{383}, respectively. We thus improved the best existing attack, which was evaluated by Liu et al. [19], against the reduced-round Friet-PC. We remark that the proposed attacks are no practical threat to Friet-PC, however, it is recommended to use these attack vectors of bit-wise cryptanalysis to evaluate the security of AND-RX ciphers when designing the AND-RX ciphers in the future.

References

[1] Tomer Ashur and Yunwen Liu. Rotational cryptanalysis in the presence of constants. *IACR Trans. Symm. Cryptol.*, 2016(1):57–70, 2016. `http://tosc.iacr.org/index.php/ToSC/article/view/535`.

[2] Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of Salsa, ChaCha, and Rumba. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 470–488, Lausanne, Switzerland, February 10–13, 2008. Springer, Heidelberg, Germany.

[3] Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. `https://131002.net/data/papers/AM09.pdf`.

[4] Stefano Barbero, Emanuele Bellini, and Rusydi Makarim. Rotational Analysis of ChaCha Permutation. *IACR Cryptology ePrint Archive*, 2020:1049, 2020.

[5] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337, Toronto, Ontario, Canada, August 11–12, 2012. Springer, Heidelberg, Germany.

[6] Arka Rai Choudhuri and Subhamoy Maitra. Significantly improved multi-bit differentials for reduced round Salsa and ChaCha. *IACR Trans. Symm. Cryptol.*, 2016(2):261–287, 2016. `http://tosc.iacr.org/index.php/ToSC/article/view/574`.

[7] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *FSE'97*, volume 1267 of *LNCS*, pages 149–165, Haifa, Israel, January 20–22, 1997. Springer, Heidelberg, Germany.

[8] Magnus Daum. *Cryptanalysis of Hash functions of the MD4-family*. PhD thesis, Ruhr-Universität Bochum, Universitätsbibliothek, 2005.

[9] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Markus Schofnegger. Algebraic cryptanalysis of variants of frit. In *SAC 2019*, LNCS, pages 149–170, Calgary, Alberta, Canada, 2019. Springer, Heidelberg, Germany.

[10] Jian Guo, Pierre Karpman, Ivica Nikolic, Lei Wang, and Shuang Wu. Analysis of BLAKE2. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 402–423, San Francisco, CA, USA, February 25–28, 2014. Springer, Heidelberg, Germany.

[11] Gurobi Optimization Inc. Gurobi optimizer 9.0, 2019. `http://www.gurobi.com/`.

[12] Ryoma Ito. Rotational Cryptanalysis of Salsa Core Function. In Willy Susilo, Robert H. Deng, Fuchun Guo, Yannan Li, and Rolly Intan, editors, *23rd International Conference on Information Security - ISC 2020*, volume 12472 of *Lecture Notes in Computer Science*, pages 129–145, Cham, 2020. Springer International Publishing.

[13] Dmitry Khovratovich and Ivica Nikolic. Rotational cryptanalysis of ARX. In Seokhie Hong and Tetsu Iwata, editors, *FSE 2010*, volume 6147 of *LNCS*, pages 333–346, Seoul, Korea, February 7–10, 2010. Springer, Heidelberg, Germany.

[14] Dmitry Khovratovich, Ivica Nikolic, Josef Pieprzyk, Przemyslaw Sokolowski, and Ron Steinfeld. Rotational cryptanalysis of ARX revisited. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 519–536, Istanbul, Turkey, March 8–11, 2015. Springer, Heidelberg, Germany.

[15] Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational rebound attacks on reduced Skein. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 1–19, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[16] Liliya Kraleva, Tomer Ashur, and Vincent Rijmen. Rotational Cryptanalysis on MAC Algorithm Chaskey. *IACR Cryptology ePrint Archive*, 2020:538, 2020.

[17] Susan K. Langford and Martin E. Hellman. Differential-linear cryptanalysis. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 17–25, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany.

[18] Yunwen Liu, Glenn De Witte, Adrián Ranea, and Tomer Ashur. Rotational-XOR cryptanalysis of reduced-round SPECK. *IACR Trans. Symm. Cryptol.*, 2017(3):24–36, 2017.

[19] Yunwen Liu, Siwei Sun, and Chao Li. Rotational Cryptanalysis From a Differential-linear Perspective: Practical Distinguishers for Round-reduced FRIET, Xoodoo, and Alzette. *IACR Cryptology ePrint Archive*, 2021:189, 2021.

[20] Jinyu Lu, Yunwen Liu, Tomer Ashur, Bing Sun, and Chao Li. Rotational-XOR Cryptanalysis of Simon-like Block Ciphers. *IACR Cryptology ePrint Archive*, 2020:486, 2020.

[21] Itsik Mantin and Adi Shamir. A practical attack on broadcast RC4. In Mitsuru Matsui, editor, *FSE 2001*, volume 2355 of *LNCS*, pages 152–164, Yokohama, Japan, April 2–4, 2002. Springer, Heidelberg, Germany.

[22] Pawel Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced Keccak. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 241–262, Singapore, March 11–13, 2014. Springer, Heidelberg, Germany.

[23] Lingyue Qin, Xiaoyang Dong, Keting Jia, and Rui Zong. Key-dependent cube attack on reduced Frit permutation in duplex-AE modes. Cryptology ePrint Archive, Report 2019/170, 2019. `https://eprint.iacr.org/2019/170`.

[24] Zhenqing Shi, Bin Zhang, Dengguo Feng, and Wenling Wu. Improved key recovery attacks on reduced-round Salsa20 and ChaCha. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC 12*, volume 7839 of *LNCS*, pages 337–351, Seoul, Korea, November 28–30, 2013. Springer, Heidelberg, Germany.

[25] Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, and Niels Samwel. Towards lightweight cryptographic primitives with built-in fault-detection. Cryptology ePrint Archive, Report 2018/729, 2018. `https://eprint.iacr.org/2018/729`.

[26] Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, and Niels Samwel. Friet: An authenticated encryption scheme with built-in fault detection. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 581–611. Springer, Heidelberg, Germany, May 2020.

[27] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 357–377, Bochum, Germany, March 20–23, 2016. Springer, Heidelberg, Germany.

[28] 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, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 648–678, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.