IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2018, No. 1, pp. 57–73. DOI:10.13154/tosc.v2018.i1.57-73

Distinguishing Attack on NORX Permutation

Tao Huang and Hongjun Wu

School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
huangtao@ntu.edu.sg, wuhj@ntu.edu.sg

Abstract. NORX is a permutation-based authentication scheme which is currently a third-round candidate of the ongoing CAESAR competition. The security bound of NORX is derived from the sponge construction applied to an ideal underlying permutation. In this paper, we show that the NORX core permutation is non-ideal with a new distinguishing attack. More specifically, we can distinguish NORX64 permutation with $2^{48.5}$ queries and distinguish NORX32 permutation with $2^{64.7}$ queries using carefully crafted differential-linear attacks. We have experimentally verified the distinguishing attack on NORX64 permutation. Although the distinguishing attacks reveal the weakness of the NORX permutation, it does not directly threat the security of the NORX authenticated encryption scheme.

Keywords: NORX · Distinguishing Attack · Differential-Linear Cryptanalysis

1 Introduction

Confidentiality and integrity are two main security notions of symmetric-key cryptography. Authenticated encryption (AE) or extended authenticated encryption with associated data (AEAD) schemes are widely used to achieve both confidentiality and integrity. The ongoing Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [CAE13] is a competition on designing authenticated encryption schemes which are better than current widely-used AE scheme AES-GCM. There are 57 algorithms submitted to the first round of this competition in 2013. In August 2016, 15 algorithms have been selected in the third round.

NORX [AJN16] is an authenticated cipher designed by Aumasson, Jovanovic and Neves. It was submitted to the CAESAR competition, and has been selected as 1 of the 15 third-round candidates. NORX is designed for efficient implementations in both software and hardware. It is an application of the monkeyDuplex construction [BDPA12, BDPA11] which uses a permutation as the underlying primitive to achieve authenticated encryption. The core permutation of NORX has a ChaCha-like round function which replaces the modular addition in ChaCha with AND, XOR and SHIFT operations. NORX supports both 32-bit word size and 64-bit word size, which are denoted as NORX32 and NORX64 respectively.

NORX has three versions so far. NORX v1.0 [AJN14] is the version in the initial submission to the CAESAR competition. NORX v2.0 [AJN15b] is the tweaked version in the second-round competition which mainly increases the rate of NORX v1.0. NORX v3.0 [AJN16] is the latest version submitted for the third-round competition with a number of tweaks to enhance its security in initialisation and finalisation.

Related work. Since the publication of NORX, there have been quite a few analyses on its security.

Received: 2017-11-23, Accepted: 2018-01-23, Published: 2018-03-01

The initial security analysis by the designers showed that high probability differentials can not be used to attack the NORX AEAD scheme. Meanwhile, the algebraic cryptanalysis and rotational cryptanalysis are difficult to be used to attack NORX.

Aumasson et al. [AJN15a] analysed the differential property of the NORX core permutation when differences can only be introduced in the nonce during initialisation. In that case, they provided upper bounds for the differential probabilities of 1 round NORX64 and NORX32 permutations, which are 2^{-53} and 2^{-60} respectively. They also studied the 4-round case, in which the differential probabilities are 2^{-584} (for NORX32) and 2^{-836} (for NORX64).

Das et al. [DMM15] analysed the higher order differential properties of the NORX core permutation. They constructed probabilistic zero-sum distinguishers for 4-round NORX64 permutation and 3.5-round NORX32 permutation. The 4-round distinguisher was constructed by choosing intermediate states, computing 4-th order differential backward for 2.25 rounds and forward by 1.75 rounds.

Bagheri et al. [BHJ+16] presented state/key recovery attacks for both NORX32 and NORX64 when the round number is reduced to 2 (out of 4). Their attacks exploit the slow diffusion of the $\mathtt{G}^{-1}$ function and make use of the internal differential in the parallel mode of NORX. They also presented a practical 2-round differential-linear distinguisher for NORX64 in the parallel mode.

Chaigneau et al. [CFG+17] proposed an attack on the full primitive of NORX v2.0. The attack exploited a structural property that the 4 columns are rotationally identical in NORX permutation. The security of NORX v2.0 was reduced to 66 bits for NORX32 and 130 bits for NORX64. This attack does not work for NORX v3.0 since the secret key is XORed with the state before finalisation.

Recently, Biryukov et al. [BUV17] analysed the NORX core permutation using a new type of truncated differentials called symmetric truncated differentials, which resulted in 2.125 rounds distinguishers for both NORX32 and NORX64.

The previous analyses on NORX permutation can be categorised into two types. The first type is the structural distinguisher, which exploits the weakness that the same operations are applied to each column of the state in NORX round function. This type of distinguisher works for any number of rounds, but can be tweaked by adding non-symmetric operation in the round function. The second type is the non-structural distinguisher. Although Das et al. [DMM15] presented a full-round higher order differential distinguisher for NORX64, it has to use the intermediate state and compute partial rounds backwards and forwards. A more realistic assumption is that the adversary can make queries to an oracle which is either a random permutation or a NORX permutation. Then he needs to decide whether the oracle is a random permutation or not. Under this assumption, the attack proposed in [DMM15] only works for 2-round NORX64 permutation in the forward direction. The recent analysis by Biryukov et al. [BUV17] reached 2.125 rounds. Hence, there is still no known distinguisher for the full NORX core permutation without using the structural property.

Our contribution. In this paper, we present our cryptanalysis against NORX64 permutation and NORX32 permutation. We show that the full NORX64 permutation can be distinguished with $2^{48.5}$ queries and the full NORX32 permutation can be distinguished with $2^{64.7}$ queries using carefully crafted differential-linear characteristics. We have experimentally verified the distinguishing attack on NORX64 permutation.

Unlike the results proposed in [CFG+17], the distinguishing attacks presented in our paper are not structural but use differential-linear characteristics. Hence, our analysis demonstrates a second time that the NORX permutations do not meet the security assumption for the underlying permutation of the monkeyDuplex construction.

To the best of our knowledge, this is the first non-structural distinguisher for NORX32

permutation. If we consider the permutation as a whole, it is also the first non-structural distinguisher for NORX64 permutation.

Outline. The rest of this paper is structured as follows. The specification of NORX is given in Section 2. Section 3 describes a differential-linear distinguishing attack on NORX64 permutation. Section 4 presents the distinguishing attack on NORX32. Section 5 concludes the paper.

2 Preliminaries

2.1 Notation

In this paper, we mostly follow the notations used in the NORX specification [AJN16]. The list of notations is given below.

w	NORX word size, either 32-bit or 64-bit string.
S	The state of NORX.
s_i	The i-th word of NORX state.
$s_i[j]$	The j-th bit of s_i. Negative j refers to ($j \mod w$)-th bit.
K	The secret key.
$x \ll n$	Left-shift of bitstring x by n bits.
$x \ggg n$	Right-rotation of bitstring x by n bits.
$\wedge$	Bitwise AND.
$\oplus$	Bitwise XOR.

Note that little-endian is used in NORX.

2.2 Specification of NORX

NORX [AJN16] is an authenticated cipher based on the monkeyDuplex construction [BDPA12]. Five instances are specified in the latest version NORX v3.0. The first four instances use the standard mode while the last instance uses the parallel mode. In the four standard instances, the first two instances use 4-round permutation and the other two instances use 6-round permutation. We will focus on the first two instances in this paper. The instance with 32-bit word size, or 512-bit state size, is denoted by NORX32. The instance with 64-bit word size, or 1024-bit state size, is denoted by NORX64. A layout of standard NORX construction is represented in Fig. 1.

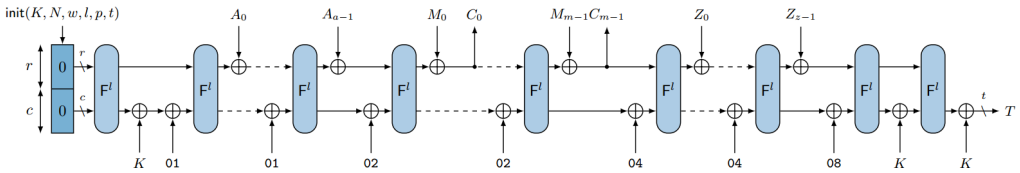

Figure 1: The layout of NORX construction (from [AJN16]). $\mathtt{F}^l$ denotes an l-round permutation of NORX.

The NORX state S consists of 16 words $s_0, \ldots, s_{15}$. Each word is w bits. The 16 state words are arranged in a 4×4 matrix:

$$\begin{bmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{bmatrix}$$

The 12 words $s_0, \ldots, s_{11}$ are the rate words in the sponge construction and the 4 words $s_{12}, \ldots, s_{15}$ are the capacity words in the sponge construction. The rate words are XORed with the message block in each iteration of the permutation. A domain separation constant is XORed with the capacity.

In this paper, we are only interested in analysing the NORX core permutation. Thus, we omit the details of the NORX AEAD mode. More details of the NORX specification can be found in [AJN16].

We use $\mathtt{F}$ to denote the NORX round function. The core permutation of NORX has 4 or 6 rounds depending on the variants of parameters. In this paper, we will study the 4-round permutation, $\mathtt{F}^4$, which is the primary recommendation by the designers. The round function $\mathtt{F}$ includes the application of a function called $\mathtt{G}$ to each column of the state followed by applying it to each diagonal of the state. Hence $\mathtt{F}(s_0, \ldots, s_{15})$ consists of column steps as follows:

$$\mathtt{G}(s_0, s_4, s_8, s_{12}), \mathtt{G}(s_1, s_5, s_9, s_{13}), \mathtt{G}(s_2, s_6, s_{10}, s_{14}), \mathtt{G}(s_3, s_7, s_{11}, s_{15}),$$

followed by the following diagonal steps:

$$\mathtt{G}(s_0, s_5, s_{10}, s_{15}), \mathtt{G}(s_1, s_6, s_{11}, s_{12}), \mathtt{G}(s_2, s_7, s_8, s_{13}), \mathtt{G}(s_3, s_4, s_9, s_{14}).$$

The function $\mathtt{G}(a, b, c, d)$ computes the following 8 operations:

$$\begin{array}{llll} 1.\ a \leftarrow \mathtt{H}(a, b), & 2.\ d \leftarrow (a \oplus d) \ggg r_0, & 3.\ c \leftarrow \mathtt{H}(c, d), & 4.\ b \leftarrow (b \oplus c) \ggg r_1, \\ 5.\ a \leftarrow \mathtt{H}(a, b), & 6.\ d \leftarrow (a \oplus d) \ggg r_2, & 7.\ c \leftarrow \mathtt{H}(c, d), & 8.\ b \leftarrow (b \oplus c) \ggg r_3, \end{array}$$

where $\mathtt{H}(x, y) = (x \oplus y) \oplus \big((x \wedge y) \ll 1\big)$. The rotation offsets (r_0, r_1, r_2, r_3) are $(8, 11, 16, 31)$ for NORX32 and $(8, 19, 40, 63)$ for NORX64. The $\mathtt{G}$ function is depicted in Figure 2.

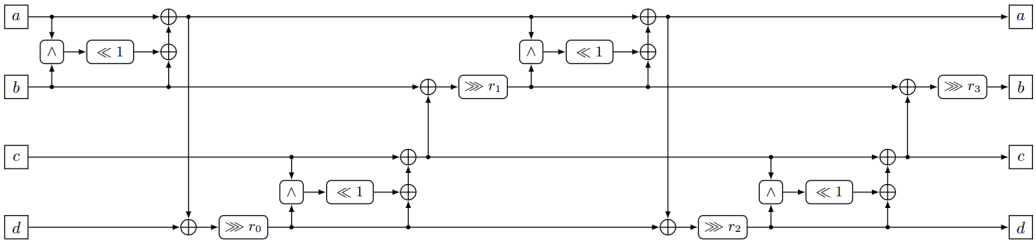

Figure 2: The $\mathtt{G}$ circuit (from [AJN16]).

In our analysis, we will use fractional number of rounds. We divide one round of NORX into two half rounds, the column half round $\mathtt{F}_{col}$ and the diagonal half round $\mathtt{F}_{diag}$. Note that the words (a, b, c, d) in the $\mathtt{G}$ function are updated twice. Thus, we can further divide the $\mathtt{F}_{col}$ and $\mathtt{F}_{diag}$ into four quarter rounds, each quarter round updates the word (a, b, c, d) once. The quarter rounds are denoted by $\mathtt{F}_{colH}$, $\mathtt{F}_{colL}$, $\mathtt{F}_{diagH}$ and $\mathtt{F}_{diagL}$ in sequence.

2.3 Distinguishing Attack

In this paper, the distinguishing attack is defined with a random oracle model. Suppose that $\mathtt{P}_{\mathrm{Rand}}$ is a random n-bit permutation and $\mathtt{P}_{\mathrm{NORX}}$ is the n-bit NORX permutation, where n is 512 or 1024. An adversary makes q queries to an oracle either a $\mathtt{P}_{\mathrm{Rand}}$ or $\mathtt{P}_{\mathrm{NORX}}$ and receive an n-bit output for each query. The goal of the adversary is to correctly identify whether the oracle is a random oracle or a NORX permutation.

3 Distinguishing Attack on NORX64 Permutation

In this section, we describe the distinguishing attack on NORX64 permutation. We start with a review of the differential-linear attack. Then, we will construct a differential-linear distinguisher step by step.

3.1 Differential-Linear Attack

The differential-linear cryptanalysis was proposed by Langford and Hellman in 1994 [LH94]. The idea of differential-linear attack is to treat the cipher as a cascade of two parts. The first part has a differential characteristic $\Delta_{in} \to \Delta_{out}$, and the second part has a linear characteristic $\Gamma_{in} \to \Gamma_{out}$. The adversary queries messages with Δ_{in} and analyse the statistics of the XORed differences of Γ_{out}. When the XORed differences have a significant bias from 0.5, the adversary can distinguish the cipher from a random permutation. In [Lu12], Lu studied the implicit assumptions made in [LH94] and [BDK02] and gave a theorem to compute the probability for the differential-linear distinguisher.

Let $\hat{p}$ be the probability that the input of the linear mask has no XORed difference after the differential step while ϵ be the linear characteristic bias for the linear step. The theorem says that the probability that the XORed difference of the output linear mask to be 0 is $\frac{1}{2} + 2(2\hat{p} - 1)\epsilon^2$. In [BLN14], Blondeau et al. further developed a method on computing the bias which only relies on the independence of the two parts of the cipher.

Note that the NORX core permutation is a key-less permutation. Thus, the independent assumption does not hold in general. We will use the $\hat{p}$ and ϵ to estimate the biases first. Then we experimentally verify and correct the estimated biases.

3.2 Constructing Linear Characteristic

Linear characteristic needs to be properly constructed to launch a differential-linear attack. We describe a linear characteristic using an input linear mask and an output linear mask. The bias of the linear characteristic is determined by the bias of the XORed sum of bits involved in the input linear mask and output linear mask.

3.2.1 Linear Approximation of the G Function

To construct a linear characteristic with large bias, we first analyse the G function in the permutation F. We will use a method similar to the one used in the analysis of ChaCha by [CM16]. First, we re-write the G function with the indexes of states. See Figure 3 for the graphic description of the linear approximation.

$$a_1 \leftarrow \mathtt{H}(a_0, b_0) \quad d_1 \leftarrow (a_1 \oplus d_0) \ggg r_0 \quad c_1 \leftarrow \mathtt{H}(c_0, d_1) \quad b_1 \leftarrow (b_0 \oplus c_1) \ggg r_1 \tag{1}$$

$$a_2 \leftarrow \mathtt{H}(a_1, b_1) \quad d_2 \leftarrow (a_2 \oplus d_1) \ggg r_2 \quad c_2 \leftarrow \mathtt{H}(c_1, d_2) \quad b_2 \leftarrow (b_1 \oplus c_2) \ggg r_3. \tag{2}$$

Then, we derive the expressions of the input a_0, b_0, c_0 and d_0 of the G function, in terms of the output a_2, b_2, c_2 and d_2. Recall that function $\mathtt{H}(x, y)$ is defined as $\mathtt{H}(x, y) = (x \oplus y) \oplus ((x \wedge y) \ll 1)$. If we replace the function $\mathtt{H}(a, b)$ with the linear approximation $x \oplus y$, the bias will be 2^{-2} for each bit except for the least significant bit (LSB) which has bias 2^{-1}. Thus, we can express any bit in the input of G as the linear combination of bits in the output of G with some bias. Let $\gamma_{x,y}[i] = x[i-1] \wedge y[i-1]$ for $i > 0$ and $\gamma_{x,y}[0] = 0$, we write the i-th bit of a_0 as:

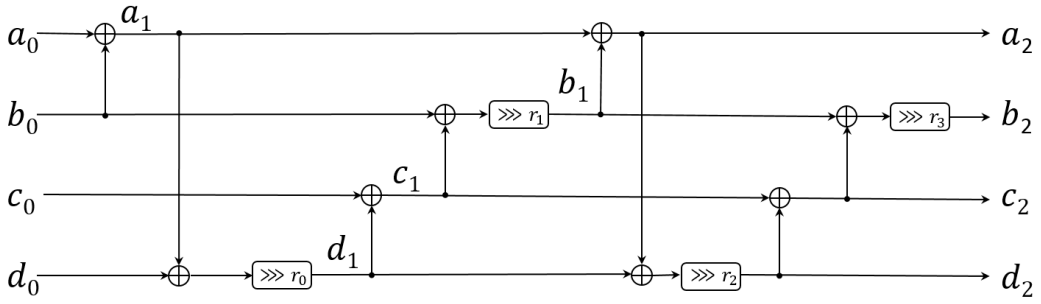

Figure 3: Linear approximation of the G function.

$$
\begin{aligned}
a_0[i] =& a_1[i] \oplus b_0[i] \oplus \gamma_{a_0,b_0}[i] \\
=& (a_2[i] \oplus b_1[i] \oplus \gamma_{a_1,b_1}[i]) \oplus (b_1[i-r_1] \oplus c_1[i]) \oplus \gamma_{a_0,b_0}[i] \\
=& a_2[i] \oplus (b_2[i-r_3] \oplus c_2[i]) \oplus \gamma_{a_1,b_1}[i] \oplus (b_2[i-r_1-r_3] \oplus c_2[i-r_1]) \oplus (c_2[i] \oplus d_2[i] \\
& \oplus \gamma_{c_1,d_2}[i]) \oplus \gamma_{a_0,b_0}[i] \\
=& a_2[i] \oplus b_2[i-r_3] \oplus b_2[i-r_1-r_3] \oplus c_2[i-r_1] \oplus d_2[i] \oplus \gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i] \oplus \gamma_{c_1,d_2}[i].
\end{aligned}
$$

Similarly, we can obtain the expressions for the i-th bit of b_0, c_0 and d_0 in terms of a_2, b_2, c_2, d_2 and γ:

$$
\begin{aligned}
b_0[i] =& b_2[i-r_1-r_3] \oplus c_2[i] \oplus c_2[i-r_1] \oplus d_2[i] \oplus \gamma_{c_1,d_2}[i], \\
c_0[i] =& a_2[i] \oplus c_2[i] \oplus d_2[i] \oplus d_2[i-r_2] \oplus \gamma_{c_0,d_1}[i] \oplus \gamma_{c_1,d_2}[i], \\
d_0[i] =& a_2[i] \oplus a_2[i-r_0] \oplus b_2[i-r_3] \oplus c_2[i] \oplus d_2[i-r_0-r_3] \oplus \gamma_{a_1,b_1}[i].
\end{aligned}
$$

From the above expressions, it is easy to observe that the LSB ($i = 0$) of a_0, b_0, c_0 and d_0 can be expressed as linear combination of a_2, b_2, c_2 and d_2. In this case, the G function is linear on those bits.

When $i > 0$, the expressions for the i-th bit of a_0, b_0, c_0 and d_0 are non-linear. We will estimate the biases using the linear approximations without the γ terms for those cases. Assuming that the input bits are uniformly and independently random, the bias of $\gamma_{x,y} = x[i-1] \wedge y[i-1] = 0$ is 2^{-2}. Thus, $b_0[i]$ and $d_0[i]$ can be linearly approximated with bias 2^{-2} since only one γ term is involved in their expressions.

The biases of the linear approximations of $a_0[i]$ and $c_0[i]$ for $i > 0$ are more complicated to estimate. Since more than one terms of γ are involved in the expressions, it might be natural to consider applying the Piling-up Lemma [Mat93]. However, when we take a close look at the expressions, it turns out that the γ_{a_0,b_0} and γ_{a_1,b_1} are not independent, which violates the assumption of Piling-up Lemma. To see this, we compare the expression of $\gamma_{a_0,b_0}[i]$ and $\gamma_{a_1,b_1}[i]$:

$$
\begin{aligned}
\gamma_{a_0,b_0}[i] =& a_0[i-1] \wedge b_0[i-1] \\
\gamma_{a_1,b_1}[i] =& a_1[i-1] \wedge b_1[i-1] \\
=& (a_0[i-1] \oplus b_0[i-1] \oplus \gamma_{a_0,b_0}[i-1]) \wedge b_1[i-1].
\end{aligned}
$$

It is clear that the bits $a_0[i-1]$ and $b_0[i-1]$ affect both γ_{a_0,b_0} and γ_{a_1,b_1}. Therefore, to accurately estimate the biases of the linear approximations of $a_0[i]$ and $c_0[i]$, we need to compute the probabilities.

First, we compute the bias for the linear approximation of $a_0[i]$ when $i > 0$. Let event A be $\gamma_{a_0,b_0} = 0$, event B be $\gamma_{a_1,b_1} = 0$. There are two cases:

Case $i = 1$. In this case, $\gamma_{a_0,b_0}[i-1] = 0$. We write the probability of $\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i] = 0$ as :

$$\mathbf{Pr}[\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i] = 0] = \mathbf{Pr}[AB] + \mathbf{Pr}[\bar{A}\bar{B}]. \tag{3}$$

Then we have

$$\begin{aligned}\mathbf{Pr}[AB] =& \mathbf{Pr}[(a_0[i-1] = 0) \wedge (b_0[i-1] = 0)] + \mathbf{Pr}[(a_0[i-1] = 0) \wedge (b_0[i-1] = 1) \\ & \wedge (b_1[i-1] = 1)] + \mathbf{Pr}[(a_0[i-1] = 1) \wedge (b_0[i-1] = 0) \wedge (b_1[i-1] = 1)] \\ =& \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \\ =& \frac{1}{2}.\end{aligned}$$

Since event $\bar{A}$ implies that both $a_0[i-1]$ and $b_0[i-1]$ are 1, we have $a_1[i-1] = a_0[i-1] \oplus b_0[i-1] = 0$. Then event B must occur, which implies $\mathbf{Pr}[\bar{A}\bar{B}] = 0$. Therefore,

$$\mathbf{Pr}[\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i] = 0] = \mathbf{Pr}[AB] = \frac{1}{2}.$$

Now we can see that the bias of $\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i]$ is 0. Since $\gamma_{c_1,d_2}[i]$ is independent of $\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i]$, the bias of $\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i] \oplus \gamma_{c_1,d_2}[i]$ is also 0, which shows the linear approximation for $a_0[i]$ is unbiased when $i = 1$. Hence, we should avoid using bit $a_0[1]$ in the input linear mask of the `G` function.

Case $i > 1$. In this case, we can still use Equation (3), but $\mathbf{Pr}[AB]$ and $\mathbf{Pr}[\bar{A}\bar{B}]$ are different. We have

$$\begin{aligned}\mathbf{Pr}[AB] =& \mathbf{Pr}[(a_0[i-1] = 0) \wedge (b_0[i-1] = 0) \wedge (\gamma_{a_0,b_0}[i-2] = 0)] \\ & + \mathbf{Pr}[(a_0[i-1] = 1) \wedge (b_0[i-1] = 0) \wedge (\gamma_{a_0,b_0}[i-2] = 1)] \\ & + \mathbf{Pr}[(a_0[i-1] = 0) \wedge (b_0[i-1] = 1) \wedge (\gamma_{a_0,b_0}[i-2] = 1)] \\ & + \mathbf{Pr}[(a_0[i-1] = 0) \wedge (b_0[i-1] = 0) \wedge (\gamma_{a_0,b_0}[i-2] = 1) \wedge (b_1[i-1] = 1)] \\ & + \mathbf{Pr}[(a_0[i-1] = 0) \wedge (b_0[i-1] = 1) \wedge (\gamma_{a_0,b_0}[i-2] = 0) \wedge (b_1[i-1] = 1)] \\ & + \mathbf{Pr}[(a_0[i-1] = 1) \wedge (b_0[i-1] = 0) \wedge (\gamma_{a_0,b_0}[i-2] = 0) \wedge (b_1[i-1] = 1)] \\ =& \frac{3}{16} + \frac{1}{16} + \frac{1}{16} + \frac{1}{32} + \frac{3}{32} + \frac{3}{32} \\ =& \frac{17}{32} \\ \mathbf{Pr}[\bar{A}\bar{B}] =& \mathbf{Pr}[(a_0[i-1] = 1) \wedge (b_0[i-1] = 1) \wedge (\gamma_{a_0,b_0}[i-2] = 1) \wedge (b_1[i-1] = 1)] \\ =& \frac{1}{32}\end{aligned}$$

Therefore,

$$\mathbf{Pr}[\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i] = 0] = \mathbf{Pr}[AB] + \mathbf{Pr}[\bar{A}\bar{B}] = \frac{9}{16}.$$

Then, the bias of $\gamma_{a_0,b_0} \oplus \gamma_{a_1,b_1}$ is $|\frac{9}{16} - \frac{1}{2}| = 2^{-4}$. The bias of $(\gamma_{a_0,b_0}[i] \oplus \gamma_{a_1,b_1}[i]) \oplus \gamma_{c_1,d_2}[i]$ can be derived from the Piling-up Lemma, which is $2 \cdot 2^{-4} \cdot 2^{-2} = 2^{-5}$.

Similarly, we can find the bias of the linear approximation for $c[i]$, which is 0 for $i = 1$ and 2^{-4} for $i > 1$. The biases of the linear approximation for 1 bit of `G` are summarised in Table 1.

With the knowledge of the linear approximation for each bit of the `G` function, the general linear approximation for `G` function can be derived. For a `G` function with input a_0, b_0, c_0 and d_0, we use Algorithm 1 to compute the linear approximation as well as the bias.

Table 1: Biases of the linear approximation for i-th bit of G function.

	$i=0$	$i=1$	$i>1$
Bias of $a[i]$	2^{-1}	0	2^{-5}
Bias of $b[i]$	2^{-1}	2^{-2}	2^{-2}
Bias of $c[i]$	2^{-1}	0	2^{-4}
Bias of $d[i]$	2^{-1}	2^{-2}	2^{-2}

Here we briefly explain how Algorithm 1 works. At a high level, it can be seen as the linear propagation of input masks to output masks. For instance, line 1 deals with the linearly approximated function $a_1 \leftarrow a_0 \oplus b_0$. Equivalently, $a_0 \leftarrow a_1 \oplus b_0$. Thus, any active bit in a_0 will propagate to both words a_1 and b_0. That is the reason we have $a_1 \leftarrow a_0$ and $b_0 \leftarrow a_0 \oplus b_0$. Line 2 to line 8 can be explained in a similar manner. Lines 13 to 17 count the number of active bits in the input, which will be used to estimate the bias. Note that the bias computation is under the assumption that the linear approximations of the input bits are mutually independent.

Algorithm 1: Linear approximation of G function.

Input : a_0, b_0, c_0, d_0 and word size w
Output : a_2, b_2, c_2, d_2 and bias ϵ

1: $a_1 \leftarrow a_0$; $b_0 \leftarrow a_0 \oplus b_0$
2: $a_1 \leftarrow d_0 \oplus a_1$; $d_1 \leftarrow d_0 \ggg r_0$
3: $c_1 \leftarrow c_0$; $d_1 \leftarrow c_0 \oplus d_1$
4: $c_1 \leftarrow b_0 \oplus c_1$; $b_1 \leftarrow b_0 \ggg r_1$
5: $a_2 \leftarrow a_1$; $b_1 \leftarrow a_1 \oplus b_1$
6: $a_2 \leftarrow d_1 \oplus a_2$; $d_2 \leftarrow d_1 \ggg r_2$
7: $c_2 \leftarrow c_1$; $d_2 \leftarrow c_1 \oplus d_2$
8: $c_2 \leftarrow b_1 \oplus c_2$; $b_2 \leftarrow b_1 \ggg r_3$
9: $cnt_a = 0$; $cnt_b = 0$; $cnt_c = 0$; $cnt_d = 0$;
10: **if** $a_0[1] = 1$ or $c_0[1] = 1$ **then**
11: $\epsilon \leftarrow 0$
12: **else**
13: **for** $i = 1, 2, \ldots, w-1$ **do**
14: $cnt_a \leftarrow cnt_a + a_0[i]$
15: $cnt_b \leftarrow cnt_b + b_0[i]$
16: $cnt_c \leftarrow cnt_c + c_0[i]$
17: $cnt_d \leftarrow cnt_d + d_0[i]$
18: **end for**
19: $\epsilon \leftarrow (-4) \cdot cnt_a + (-1) \cdot cnt_b + (-3) \cdot cnt_c + (-1) \cdot cnt_d - 1$
20: **end if**

3.2.2 Searching for Linear Characteristic

Since the linear approximations of the G function are known, we are ready to search for linear characteristics of NORX64.

We consider the case that the input of the linear characteristic has only 1 active bit. To reduce the search space, we consider the position of active bit. It is clear that bit 0 is the best choice from the analysis of the G function in Section 3.2.1. Because of the structural property of NORX permutation, we can only search the active bit in one of the four G functions applied to input state, which reduces the search space to only 4 bits, namely $a[0]$, $b[0]$, $c[0]$ and $d[0]$ of the G function. For example, if the G function is

applied to the column of state, we only need to consider the bits $s_0[0], s_4[0], s_8[0], s_{12}[0]$. Here we decompose each round of NORX64 permutation $\mathtt{F}$ into two steps: the column step $\mathtt{F}_{col}$ and the diagonal step $\mathtt{F}_{diag}$. Starting from 1 round of NORX64 permutation, $\mathtt{F}_{diag} \circ \mathtt{F}_{col}$, we compute the output masks and the corresponding biases for different rounds (incremental by half-round) by applying Algorithm 1. After 1.5 rounds, the biases for linear characteristics corresponding to $a[0]$ and $d[0]$ are 0. After 2 rounds, all the biases are vanished. The biases of the linear characteristics with 1 active bit in the input are shown in Table 2.

Table 2: Biases of the linear characteristics with 1 active bit in the input.

	$a[0]$	$b[0]$	$c[0]$	$d[0]$
$\mathtt{F}_{diag} \circ \mathtt{F}_{col}$	2^{-6}	2^{-5}	2^{-2}	2^{-7}
$\mathtt{F}_{diag} \circ \mathtt{F}_{col} \circ \mathtt{F}_{diag}$	0	2^{-29}	2^{-27}	0
$\mathtt{F}_{diag} \circ \mathtt{F}_{col} \circ \mathtt{F}_{diag} \circ \mathtt{F}_{col}$	0	0	0	0

Although the linear approximation of 1-round NORX64 permutation is significantly biased, the biases of 1.5-round NORX64 permutation are too small to be used in practical attacks. Thus, we further divide the round function NORX64 into 4 quarter rounds: $\mathtt{F}_{colH}$, $\mathtt{F}_{colL}$, $\mathtt{F}_{diagH}$, and $\mathtt{F}_{diagL}$. Now we can analyse 1.25-round NORX permutation, which is $\mathtt{F}_{diag} \circ \mathtt{F}_{col} \circ \mathtt{F}_{diagL}$. After studying different choices of the input bits, we find that when the input bit is a bit in 'c' of the $\mathtt{G}(a, b, c, d)$, or a bit in state $\{s_8, s_9, s_{10}, s_{11}\}$, the largest biased can be obtained for 1.25-round NORX64 permutation, which is 2^{-8}. For example, input bit $s_9[0]$ and output bits indicated in Table 3 form a linear characteristic for 1.25-round NORX64 permutation.

Table 3: Output bits of the linear characteristic with input $s_9[0]$.

0x0000000000000001	0x0000000000000001	0x0000000001010000	0x0000000000010101
0x0000000000020000	0x0000000000000000	0x0000c00000000002	0x0200404002000000
0x0000202001000001	0x0000000000010001	0x0000000000000001	0x0000600000000002
0x0000000000000003	0x0100010000010001	0x0000000101000001	0x0000000001000001

3.3 Constructing Differential Characteristic

In Section 3.2.2, a 1.25-round linear characteristic has been constructed. Now we construct a differential characteristic based on the linear characteristic. In a differential-linear attack, we are interested in finding a differential characteristic such that the probability that the XORed sum of difference in the bit(s) of the input linear characteristic has large bias. Since NORX permutation has 4 rounds, we need to construct a 2.75-round differential characteristic to attack the full permutation. If we consider the difference propagation from the beginning, it is difficult to control the difference after 2.75 rounds.

In our analysis on NORX64 permutation, we will construct the differential characteristic from the backward direction, so as to fit the linear characteristic. We divide the differential characteristic into three phases. Phase 1 is to propagate 1-bit difference to the 1-bit input linear mask. Phase 2 is to propagate the 1-bit difference backward probabilistically. Phase 3 is to propagate the input difference of Phase 2 backward to the input difference deterministically with initial conditions.

3.3.1 Differential Characteristic in Phase 1

In the 1.25-round linear characteristic which has been discussed in Section 3.2.2, bit $s_9[0]$ is the input and bits in Table 3 are the output. Our goal is to find a differential characteristic

with only 1 bit input difference such that the bit $s_9[0]$ in the output difference has the largest bias.

Because this phase has truncated difference in the output, we choose to experimentally search for the best differential characteristic. When 1-bit difference is considered, there are 1024 possible choices for NORX64 permutation. We generate 2^{20} pairs of random input states with 1-bit difference on the position of each state bit. Then, compute certain number of rounds of the NORX64 permutation. Note that in order to connect with the linear characteristic, the last operation should be 0.25 round $\mathtt{F}_{diagH}$. By gradually increasing the number of rounds, eventually we obtain a good differential characteristic:

$$s_{10}[17] \xrightarrow{\mathtt{F}_{diagH} \circ \mathtt{F}_{col} \circ \mathtt{F}_{diag} \circ \mathtt{F}_{col}} s_9[0]$$

with biased probability $\frac{1}{2} - 2^{-3.9}$ for 1.75-round NORX64 permutation, $\mathtt{F}_{diagH} \circ \mathtt{F}_{col} \circ \mathtt{F}_{diag} \circ \mathtt{F}_{col}$.

3.3.2 Differential Characteristic in Phase 2

Then we propagate the 1-bit difference $s_{10}[17]$ obtained in Phase 1 backwards. We simplify the non-linear operation $\mathtt{H}(a, b) = (a \oplus b) \oplus ((a \wedge b) \ll 1)$ as linear operation $a \oplus b$. Thus, the NORX64 round function becomes linear and easy to invert. Here we only need to invert propagate 0.5 round, $\mathtt{F}_{diag}$. The input difference is given in Table 4.

Table 4: Input difference in Phase 2.

0x0000001000000000	0x0000000000000000	0x0000000000000000	0x0000000000000000
0x0000000000000000	0x0000001000020000	0x0000000000000000	0x0000000000000000
0x0000000000000000	0x0000000000000000	0x0000000000020000	0x0000000000000000
0x0000000000000000	0x0000000000000000	0x0000000000000000	0x0000000000020000

Note that the probability for the linear approximation is 0.5 for each active bit except for the LSB in H function. By counting the active bits involved in the H function, the probability for the 0.5 round differential characteristic is 2^{-5}.

3.3.3 Differential Characteristic in Phase 3

After the Phase 1 and Phase 2, we only need to deal with the last 0.5 round of NORX64 permutation which is $\mathtt{F}_{col}$. Similar to Phase 2, we use linear approximation of H function to derive the differential characteristic for this 0.5 round. However, the probability of the differential characteristic in this phase is not probabilistic. By setting conditions on the initial state, the probability that the linear approximations of H function hold can be 1.

First, we derive the initial difference by propagating the input difference obtained in Phase 2 backwards. The resulting input difference in Phase 3 is shown in Table 5.

Table 5: Input difference in Phase 3.

0x0000001000000000	0x0040000000010000	0x0000001000000000	0x0000000000020000
0x0000000000000000	0x0040000800000000	0x0000001000020000	0x0000000000020000
0x0000001000000000	0x0000000000000000	0x0000000000020000	0x0000000000020000
0x0000101000000000	0x0000000800010000	0x0000000000020000	0x0000000000000002

Then, we derive the initial conditions for $\mathtt{F}_{col}$. Since the computations of the 4 columns are independent, we can derive the conditions for each column.

For Column 0, which is the left most column, we denote the operation as $\mathtt{G}(a_0, b_0, c_0, d_0)$. In this case, $(a_0, b_0, c_0, d_0) = (s_0, s_4, s_8, s_{12})$. We use the notations of intermediate states,

(a_1, b_1, c_1, d_1) and (a_2, b_2, c_2, d_2) as defined in Equation (1) and Equation (2). Now we analyse the 4 applications of non-linear function H in G.

1. $\mathbf{a_1} = \mathtt{H}(\mathbf{a_0}, \mathbf{b_0})$. The difference on a_0 is 0x0000001000000000 and the difference on b_0 is 0. The linear approximation $\mathtt{H}(a_0, b_0) = a_0 \oplus b_0$ holds with probability 1 if the following conditions holds:

$$0 = b_0[36].$$

Thus, we set a condition $b_0[36] = 0$, or $s_4[36] = 0$, to make sure the above condition holds.

2. $\mathbf{c_1} = \mathtt{H}(\mathbf{c_0}, \mathbf{d_1})$. The difference on c_0 is 0x0000001000000000 and the difference on d_1 is 0x0000001000000000. The linear approximation $\mathtt{H}(c_0, d_1) = c_0 \oplus d_1$ holds with probability 1 if the following conditions holds:

$$\begin{aligned} 1 =& c_0[36] \oplus d_1[36] \\ =& c_0[36] \oplus d_0[44] \oplus a_1[44] \\ =& c_0[36] \oplus d_0[44] \oplus a_0[44] \oplus b_0[44] \oplus (a_0[43] \wedge b_0[43]). \end{aligned}$$

We set conditions $a_0[43, 44] = 0$, $b_0[44] = 0$, $c_0[36] = 0$, and $d_0[44] = 1$ to make sure the above condition holds.

3. $\mathbf{a_2} = \mathtt{H}(\mathbf{a_1}, \mathbf{b_1})$. The difference on a_1 is 0x1000000000 and the difference on b_1 is 0. The linear approximation $\mathtt{H}(a_1, b_1) = a_1 \oplus b_1$ holds with probability 1 if the following conditions holds:

$$\begin{aligned} 0 =& b_1[36] \\ =& b_0[55] \oplus c_1[55] \\ =& b_0[55] \oplus c_0[55] \oplus d_1[55] \oplus (c_0[54] \wedge d_1[54]) \\ =& b_0[55] \oplus c_0[55] \oplus d_0[63] \oplus a_1[63] \oplus (c_0[54] \wedge d_1[54]) \\ =& b_0[55] \oplus c_0[55] \oplus d_0[63] \oplus a_0[63] \oplus b_0[63] \oplus (a_0[62] \wedge b_0[62]) \oplus (c_0[54] \wedge d_1[54]). \end{aligned}$$

We set conditions $a_0[62, 63] = 0$, $b_0[55, 63] = 0$, $c_0[54, 55] = 0$, and $d_0[63] = 0$ to make sure the above condition holds.

4. $\mathbf{c_2} = \mathtt{H}(\mathbf{c_1}, \mathbf{d_2})$. Since both c_1 and d_2 have no active bit, the linear approximation holds with probability 1.

Thus, we have derived all the conditions for Column 0. Other conditions can be derived in an analogous manner. Note that the situation can be more complicated when some bit in the initial state are involved in more than one conditions. In that case, we need to take care of all conditions to avoid contradiction on those bits. We summarise all the conditions on the initial state in Table 6. We have experimentally verified that with the initial conditions, the differential characteristic in Phase 3 holds with probability 1.

3.4 The Differential-Linear Distinguisher for NORX64 Permutation

After constructed both differential characteristic and linear characteristic, we are ready to combine them to form a differential-linear distinguisher for NORX64 permutation.

Let Δ_{in} be the difference given in Table 5 and Γ_{out} be the bits specified in the output linear mask in Table 3. The differential-linear characteristic is specified by Δ_{in} and Γ_{out}. Then we estimate the differential-linear bias.

Table 6: Conditions on the initial state for NORX64.

Column 0	$s_0[43, 44, 62, 63] = 0$, $s_4[36, 44, 55, 63] = 0$, $s_8[36, 54, 55] = 0$, $s_{12}[44] = 1$, $s_{12}[63] = 0$
Column 1	$s_1[15, 16, 34, 35, 42, 43, 54, 61, 62] = 0$, $s_5[16, 35, 43, 54, 62] = 0$, $s_9[35, 54] = 1$, $s_9[53] = 0$, $s_{13}[43, 62] = 0$
Column 2	$s_2[0, 1, 16, 17, 19, 20, 24, 25, 43, 44, 55, 56, 57] = 0$, $s_2[36] = 1$, $s_6[1, 12, 17, 20, 25, 36, 44, 56, 57] = 0$, $s_{10}[11, 12, 35] = 0$, $s_{10}[36] = 1$, $s_{14}[1, 20, 25, 44] = 0$
Column 3	$s_3[0, 1, 17, 19, 20, 24, 25, 55, 56, 57] = 0$, $s_7[1, 12, 20, 25, 56, 57] = 0$, $s_7[17] = 1$, $s_{11}[11, 12, 16] = 1$, $s_{11}[17, 57] = 1$, $s_{15}[1, 20, 25] = 0$

Let $p_0 = 2^{-5}$ be the differential probability in Phase 2, and $p_1 = 1/2 - 2^{-3.9}$ be the differential probability in Phase 1. We assume that when the difference at the beginning of Phase 1 is other than the bit $s_{10}[17]$, the probability that the bit $s_9[0]$ has difference is unbiased with probability $\frac{1}{2}$. Thus the estimated differential probability $\hat{p}$ for $(\Delta_{in} \to \Delta_{out})$ is $p_0p_1+(1-p_0)\cdot\frac{1}{2} = \frac{1}{2}+p_0(p_1-\frac{1}{2}) = \frac{1}{2}+2^{-5}(\frac{1}{2}-2^{-3.9}-\frac{1}{2}) = \frac{1}{2}+2^{-5}\cdot(-2^{-3.9}) = \frac{1}{2}-2^{-8.9}$. Let Γ_{in} be the bit $s_9[0]$ and Γ_{out} be the bits given in Table 3, the bias ϵ of the linear approximation $\Gamma_{in} \to \Gamma_{out}$ is 2^{-8}. Hence, the differential-linear bias is given by $2(2\hat{p}-1)\epsilon^2$, which is $-2^{-22.9}$.

The distinguishing attack can be performed in the following procedure.

1. Query $2^{47.5}$ pairs of 1024-bit message with difference Δ_{in} and initial conditions in Table 6.
2. For each pair of output from the oracle, compute the XORed sum of bits in Γ_{out}.
3. Count the number X that is the number of pairs such that the XORed sum is 0.
4. If $X < 2^{46.5} - 2^{23.6}$, the oracle is the NORX64 permutation. Otherwise, the oracle is a random permutation.

Now we compute the success probability of the above distinguishing attack. When the oracle is the NORX64 permutation, the random variable X can be approximated by a normal distribution $X \sim N(2^{47.5}p, \sqrt{2^{47.5}p(1-p)})$, where $p = \frac{1}{2} - 2^{-22.9}$. When the oracle is a random permutation, the random variable X can be approximated by a normal distribution $X \sim N(2^{47.5}/2, \sqrt{2^{47.5}/4})$. Thus, the probability that $X < 2^{46.5} - 2^{23.6}$ is larger than 96% when the oracle is the NORX64 permutation. The probability that $X > 2^{46.5} - 2^{23.6}$ is larger than 96% when the oracle is a random permutation.

3.5 Experimental Results

Since the complexity of the distinguishing attack on NORX64 permutation is relatively low, we can experimentally perform the distinguishing attack, which verifies the estimated differential-linear bias and the complexity of the attack.

We generate $2^{47.49}$ pairs of random input with the initial condition and difference specified by the differential-linear characteristic in Section 3.4. It takes 63.1 hours on a

GPU server with 4 Tesla K-40 GPUs to count the sum of XORed output linear mask. The bias on the output bits is $-2^{-22.88}$, which is very close to the estimated $-2^{22.9}$. This clearly distinguishes the NORX64 permutation from a random permutation since the probability that a random permutation has such bias is only 3%.

4 Distinguishing Attack on NORX32 Permutation

In this section, we will study the NORX32 permutation. It turns out a similar differential-linear distinguishing attack can be applied to NORX32 permutation. Like the previous, section, we will start with constructing the linear characteristic and then construct the differential characteristic backward to form a differential-linear characteristic for NORX32 permutation.

4.1 Constructing Linear Characteristic

The construction of linear characteristic is almost the same as NORX64, except different word size and rotation constants are used in this case. Thus, we still choose the input linear mask Γ_{in} to be the bit $s_9[0]$. Then, the output linear mask Γ_{out} after 1.25 rounds is obtained by the linear approximation for each round function. The output linear mask is shown in Table 7.

Table 7: Output bits of the linear characteristic with input $s_9[0]$.

0x00000001	0x00000001	0x00010100	0x00000100
0x00000200	0x00000000	0x00c00002	0x02424000
0x00212001	0x00000101	0x00000001	0x00600002
0x00000003	0x00000101	0x00000001	0x00010001

The estimated bias for the above linear approximation is 2^{-15} which is the same as for NORX64.

4.2 Constructing Differential Characteristic

To construct the differential characteristic for NORX32 permutation, we will use the 3 phases method in Section 3.3. However, as the differential propagation in NORX32 is different from NORX64, we need to adjust the number of rounds in Phase 1 and Phase 2.

4.2.1 Differential Characteristic in Phase 1

To find the differential characteristic in Phase 1, we search the biased probabilities that 1-bit input difference propagate to $s_9[0]$ for different number of rounds. Unlike the Phase 1 differential characteristic in NORX64, we can only find highly biased probability differential for 1.5 rounds. This 1.5 rounds differential characteristic is given by:

$$s_{11}[5] \xrightarrow{\mathtt{F}_{diagH} \circ \mathtt{F}_{col} \circ \mathtt{F}_{diag} \circ \mathtt{F}_{colL}} s_9[0]$$

with probability $\frac{1}{2} - 2^{-4.2}$.

4.2.2 Differential Characteristic in Phase 2

In Phase 2, we propagate the difference $s_{11}[5]$ backward for 0.75 round, $\mathtt{F}_{colH} \circ \mathtt{F}_{dial}$. The resulting difference is given in Table 8.

The differential probability in this phase is 2^{-13}.

Table 8: Input difference in Phase 2.

0x00000000	0x00010000	0x00008010	0x00000020
0x00000000	0x00000000	0x00010020	0x00008000
0x00000000	0x00000020	0x00000000	0x00000020
0x00000020	0x00000010	0x00002020	0x00000000

4.2.3 Differential Characteristic in Phase 3

In this phase, we first derive the input difference in the initial state by inverting the input difference in Phase 2 for another 0.5 round, $\mathtt{F}_{col}$. The resulting difference is given in Table 9.

Table 9: Input difference in Phase 2.

0x00000020	0x00000010	0x0400a020	0x02014020
0x00000020	0x00010030	0x0400a020	0x02010020
0x00200020	0x00110030	0x2020a030	0x00000000
0x20000000	0x11010020	0x20801020	0x00006000

Although the number of active involved in the non-linear function is much higher than the NORX64 case, we are still able to derive a set of conditions on the initial state to ensure the Phase 1 has differential probability 1. The conditions are listed in Table 10.

Table 10: Conditions on the initial state for NORX32.

Column 0	$s_0[5] = 1,\ s_0[7, 8, 12, 13, 19, 20, 21, 28, 29] = 0,$ $s_4[0, 5, 8, 13, 20, 21, 29] = 0,$ $s_8[0, 4] = 0,\ s_8[5, 21] = 1,$ $s_{12}[8, 13, 29] = 0$
Column 1	$s_1[2, \ldots, 8, 11, 12, 13, 16, 18, 19, 20, 21, 23, 24, 27, 28, 29] = 0,$ $s_5[0, 3, 5, 7, 8, 12, 13, 16, 19, 20, 21, 24, 27, 28] = 0,\ s_5[4] = 1,$ $s_9[0, 3, 15, 26, 27, 30, 31] = 0,\ s_9[4, 16, 20] = 1,$ $s_{13}[3, 7, 8, 12, 13, 24, 28, 29] = 0$
Column 2	$s_2[3, 4, 5, 7, 8, 11, 12, 13, 15, 16, 19, \ldots, 23, 26, 28, 29] = 0,\ s_2[14] = 1,$ $s_6[0, 4, 8, 12, 16, 20, 21, 23, 28, 29] = 0,\ s_6[5, 13, 14, 26] = 1,$ $s_{10}[0, 5, 7, 8, 12, 13, 21, 29, 31] = 0,\ s_{10}[4, 15] = 1,$ $s_{14}[5, 8, 12, 16, 21, 23] = 0,\ s_{14}[13, 29] = 1$
Column 3	$s_3[0, 1, 5, 7, 8, 13, 14, 16, 19, 20, 21, 25, 28, 29] = 0,\ s_3[4] = 1,$ $s_7[0, 1, 8, 14, 20, 21, 29] = 0,\ s_7[4, 5, 16, 25] = 1,$ $s_{11}[0, 5, 24, 25, 31] = 0,$ $s_{15}[1, 8, 29] = 0$

4.3 The Differential-Linear Distinguisher for NORX32 Permutation

Now we can construct a differential-linear distinguisher for NORX32 permutation.

Let Δ_{in} be the difference given in Table 9 and Γ_{out} be the bits specified in the output linear mask in Table 7. The differential-linear characteristic is specified by Δ_{in} and Γ_{out}. Then we estimate the differential-linear bias.

The estimated differential probability $\hat{p}$ for $(\Delta_{in} \to \Delta_{out})$ is $\frac{1}{2} + 2^{-13} \cdot (-2^{-4.2}) = \frac{1}{2} - 2^{-17.2}$. Let Γ_{in} be the bit $s_9[0]$ and Γ_{out} be the bits given in Table 7, the bias ϵ of

the linear approximation $\Gamma_{in} \to \Gamma_{out}$ is 2^{-8}. Hence, the differential-linear bias is given by $2(2\hat{p}-1)\epsilon^2$, which is $-2^{-31.2}$.

The distinguishing attack can be performed in the following procedure.

1. Query $2^{63.7}$ pairs of 512-bit message with difference Δ_{in} and initial conditions in Table 10.
2. For each pair of output from the oracle, compute the XORed sum of bits in Γ_{out}.
3. Count the number X that is the number of pairs such that the XORed sum is 0.
4. If $X < 2^{62.7} - 2^{31.7}$, the oracle is the NORX32 permutation. Otherwise, the oracle is a random permutation.

Now we compute the success probability of the above distinguishing attack. When the oracle is the NORX32 permutation, the random variable X can be approximated by a normal distribution $X \sim N(2^{63.7}p, \sqrt{2^{63.7}p(1-p)})$, where $p = \frac{1}{2} - 2^{-31.2}$. When the oracle is a random permutation, the random variable X can be approximated by a normal distribution $X \sim N(2^{63.7}/2, \sqrt{2^{63.7}/4})$. Thus, the probability that $X < 2^{62.7} - 2^{31.7}$ is larger than 96% when the oracle is the NORX32 permutation. The probability that $X > 2^{62.7} - 2^{31.7}$ is larger than 96% when the oracle is a random permutation.

5 Conclusion

The distinguishing attacks on the NORX core permutation require chosen bits in every word of the input NORX state and known bits in 15 (out of 16) words of the output NORX state. In NORX AEAD scheme, the input state of the permutation is not fully controlled by the attacker. In the initialisation, only the nonce, 4 (out of 16) words, can be controlled. In the message processing, only the rate part, 12 (out of 16) words can be controlled. Thus, the distinguishing attacks on the permutation do not lead to a real attack on NORX authenticated encryption scheme.

The security proof of NORX is based on the NORX-mode of operation with ideal underlying permutation. We study the security of the NORX core permutation for both NORX32 and NORX64 using the differential-linear cryptanalysis. It turns out that the NORX64 permutation can be distinguished with $2^{48.5}$ queries which has been experimentally verified by us. The NORX32 permutation can be distinguished with $2^{64.7}$ queries, which may be considered as semi-practical. We hope that the observations in this paper will be useful for the designers and researchers who are interested in analysing the security of NORX.

For future work, it is possible to improve the attacks in this paper by carefully selecting initial conditions to control the difference propagation for more rounds.

Acknowledgment

We would like to thank the anonymous reviewers for their helpful comments.

References

[AJN14] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX V1, 2014. http://competitions.cr.yp.to/round1/norxv1.pdf.

[AJN15a] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. Analysis of NORX: investigating differential and rotational properties. In Diego F. Aranha

and Alfred Menezes, editors, *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 306–324. Springer, 2015.

[AJN15b] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX V2.0, 2015. http://competitions.cr.yp.to/round2/norxv20.pdf.

[AJN16] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX V3.0, 2016. http://competitions.cr.yp.to/round3/norxv30.pdf.

[BDK02] Eli Biham, Orr Dunkelman, and Nathan Keller. Enhancing differential-linear cryptanalysis. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, pages 254–266, 2002.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 320–337, 2011.

[BDPA12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. *Directions in Authenticated Ciphers*, 2012.

[BHJ+16] Nasour Bagheri, Tao Huang, Keting Jia, Florian Mendel, and Yu Sasaki. Cryptanalysis of reduced NORX. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 554–574, 2016.

[BLN14] Céline Blondeau, Gregor Leander, and Kaisa Nyberg. Differential-linear cryptanalysis revisited. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 411–430, 2014.

[BUV17] Alex Biryukov, Aleksei Udovenko, and Vesselin Velichkov. Analysis of the NORX core permutation. *IACR Cryptology ePrint Archive*, 2017:34, 2017.

[CAE13] CAESAR: Competition for Authenticated Encryption: Security, Applicability,and Robustness, 2013. http://competitions.cr.yp.to/caesar.html.

[CFG+17] Colin Chaigneau, Thomas Fuhr, Henri Gilbert, Jérémy Jean, and Jean-René Reinhard. Cryptanalysis of NORX v2.0. *IACR Trans. Symmetric Cryptol.*, 2017(1):156–174, 2017.

[CM16] Arka Rai Choudhuri and Subhamoy Maitra. Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha. *IACR Trans. Symmetric Cryptol.*, 2016(2):261–287, 2016.

[DMM15] Sourav Das, Subhamoy Maitra, and Willi Meier. Higher order differential analysis of NORX. *IACR Cryptology ePrint Archive*, 2015:186, 2015.

[LH94] Susan K. Langford and Martin E. Hellman. Differential-linear cryptanalysis. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 17–25, 1994.

[Lu12] Jiqiang Lu. A methodology for differential-linear cryptanalysis and its applications - (extended abstract). In *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, pages 69–89, 2012.

[Mat93] 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.