Full Round Distinguishing and Key-Recovery Attacks on SAND-2 (Full version)

Zhuolong Zhang[1], Shiyao Chen[2], Wei Wang[1,3,4(✉)], and Meiqin Wang[1,3,4]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
zhuolongzhang@mail.sdu.edu.cn, {weiwangsdu, mqwang}@sdu.edu.cn
[2] Nanyang Technological University, Singapore, Singapore
shiyao.chen@ntu.edu.sg
[3] Quan Cheng Laboratory, Jinan, China
[4] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China

Abstract. This paper presents full round distinguishing and key recovery attacks on lightweight block cipher SAND-2 with 64-bit block size and 128-bit key size, which appears to be a mixture of the AND-Rotation-XOR (AND-RX) based ciphers SAND and ANT. However, the security arguments against linear and some other attacks are not fully provided. In this paper, we find that the combination of a SAND-like nibble-based round function and ANT-like bit-based permutations will cause dependencies and lead to iterative linear and differential trails with high probabilities. By exploiting these, full round distinguishing attacks on SAND-2 work with 2^{46} queries for linear and $2^{58.60}$ queries for differential in the single-key setting. Then, full round key recovery attacks are also mounted, which work with the time complexity $2^{48.23}$ for linear and $2^{64.10}$ for differential. It should be noted that the dependency observed in this paper only works for SAND-2 and will not threaten SAND and ANT. From the point of designers, our attacks show the risk of mixing the parts of different designs, even though each of them is well-studied to be secure.

Keywords: Linear Cryptanalysis · Differential Cryptanalysis · Distinguishing Attack · Key Recovery Attack · SAND-2

1 Introduction

With strong demands of lightweight symmetric-key primitives, the design and cryptanalysis of lightweight ciphers (*e.g.,* block cipher and hash function) has been one of the most productive lines of research in recent years. As one of the most important building blocks of symmetric primitives, lightweight block cipher has motivated and inspired many important research directions and works.

Taking a variety of cost metrics into considerations under lightweight scenarios, it is naturally a challenge to balance different perspectives when designing the block cipher, including security level, hardware cost and software efficiency.

For instance, SIMON and SPECK proposed by NSA [BSS+13] are two quite elegant and competitive algorithms but without any design rationale and security analysis in the design paper, where the former is hardware-oriented and the latter is software-oriented. SKINNY [BJK+16] is then proposed by Beierle *et al.* at CRYPTO 2016 as a competitor to SIMON in terms of performance, and it provides stronger security guarantees with regard to differential [BS91] and linear [Mat93] attacks, which are the most classical and powerful cryptanalytic methods. Later, Chen *et al.* [CFS+22] proposed a new family of AND-RX block ciphers SAND at DCC 2022, which admits an equivalent nibble-based structure, this makes SAND both software and hardware efficient. They also introduced a novel approach to analyze the security, which allows for high security in both single-key and related-key [Knu91,Bih94] scenarios.

Recently, Chen and Li *et al.* [CLGH23] follows SAND and ANT [CFF+19] block ciphers to design a new cipher called SAND-2[1], which adopts almost all SAND cipher and bit-based permutations similar to that in ANT cipher. They aim to achieve a better diffusion and security bounds of differential, however, the designers only evaluate the resistance against differential attack and do not provide other common cryptanalysis. Especially considering that the bit-based permutations totally break the nibble-based structure, it seems that the designers of SAND-2 did not take care of the dependency existing in the round function, which has already been discussed by the designers of SAND [CFS+22, Section 3]. And it is worth noting that this similar dependency has already been observed by Sasaki [Sas18] to break full round ANU cipher [BPSP16] under related-key setting. Naturally, we wonder that *whether there are some dependencies in SAND-2*? and if there exists, *whether we can make full use of such dependencies to provide more in-depth security evaluations of SAND-2*?

Contributions. In this paper, we answer the above two questions positively. By carefully observing the bit-based round function of SAND-2, we firstly find some dependencies that can be used to construct linear and differential trails, which help us derive two-round iterative differential and linear characteristics. Then, we mount longer number of rounds distinguishers from these iterative trails. Based on which, we finally launch full round key recovery attacks on SAND-2, our results are given in Table 1.

Dependencies in the round function of SAND-2. Although the designers of SAND-2 adopted bit-based permutations P_0/P_1 to mix bits in G_0/G_1 as complicated as possible, considering the software implementation, it still preserves some properties (like the partial rotation invariant property of P_0 and P_1 in ANT cipher). Based on these properties, for SAND-2 round function, we can easily derive the same input bit for two parallel non-linear components G_0 and G_1 after regrouping by rotations and bit permutations P_0/P_1.

[1] SAND-2 uses the name of SAND, but it is designed by totally different designers.

Iterative linear and differential trails of SAND-2. Based on the observed dependencies, we then construct two-round iterative linear and differential characteristics of SAND-2. For linear $(0x0, 0x2) \xrightarrow{2r} (0x0, 0x2)$, it has the linear bias with 2^{-2}. For differential $(0x8, 0x0) \xrightarrow{2r} (0x8, 0x0)$, it has the differential probability with 2^{-3}. Both only have one active bit and have other seven rotation equivalent trails due to the partial rotation invariant of bit-based permutations.

Then, we mount longer linear and differential distinguishers based on these iterative characteristics, the clustering effect is also considered, SAT/SMT based automatic search method clustering and experiments of these distinguishers are performed as verifications. For linear, these iterative based longer number of rounds distinguishers has no significant clustering effect. For differential, we develop a formula based method to approximately evaluate the clustering differential probability, especially for longer number of rounds where SAT/SMT based method is inefficient. The experiment results show that our method is effective and efficient to approximate these iterative differential distinguishers.

Full round attacks on SAND-2. With these carefully constructed and evaluated iterative trails, full round distinguishers can be mounted: $(0x2, 0x0) \xrightarrow{47r} (0x0, 0x2)$ for linear with linear probability 2^{-46} and $(0x0, 0x8) \xrightarrow{47r} (0x8, 0x0)$ for differential with differential probability $2^{-58.60}$. These distinguishers not only lead to full round distinguishing attacks on SAND-2, but also they have a high probability, especially for linear with a practical complexity 2^{46}. Then, we launch linear and differential *full round key recovery attacks* on SAND-2, which are summarized in Table 1. It is worth noting that the time and data complexity of linear full round key recovery attack are even *practical* (both under 2^{50}).

Table 1: Distinguishing and key recovery attacks on SAND-2 (the total rounds of SAND-2 are 47).

Attack Method	Rounds	Time[†]	Data	Memory	Success Prob.	Source
Distinguisher						
Differential	6	–	–	–	–	[CLGH23]
Linear	47	$2^{46.00}$	$2^{46.00}$	1	–	Section 3.1
Differential	47	$2^{58.60}$	$2^{58.60}$	1	–	Section 3.2
Key Recovery						
Linear	47 (41)[‡]	$2^{48.23}$	$2^{45.50}$	$2^{35.00}$	83.24%	Section 4.1
Differential	47 (43)	$2^{64.10}$	$2^{60.20}$	$2^{57.20}$	92.61%	Section 4.2
Differential	47 (41)	$2^{73.70}$	$2^{53.13}$	$2^{53.13}$	90.65%	Appendix A

[†] Time complexity is evaluated by one full round encryption of SAND-2.
[‡] The number of rounds of the distinguisher used for key recovery attack are 41.

Outline of the paper. In Section 2, we firstly give a brief introduction of SAND and SAND-2 block ciphers. In Section 3, we show the dependencies in the round function of SAND-2, which are used to construct differential and linear distinguishers. Based on the distinguishers we mount, full round linear and differential key recovery attacks on SAND-2 are provided in Section 4. Finally, we conclude the paper.

2 Preliminary

The design of SAND-2 almost follows SAND block cipher, except that the designers of SAND-2 break the nibble-based equivalent structure of SAND by adopting bit-based permutations in the middle of two parallel expanding round functions, which may incur dependency problems as discussed in the design rationales of SAND [CFS+22, Section 3]. This also makes the security analysis of SAND-2 more difficult, that is, the cipher cannot be analyzed under nibble-level like SAND. We note that SAND-2 still lacks enough cryptanalysis and the designers only provide a rough security bound against differential attack. Since SAND-2 directly uses the same key schedule and the similar round function of SAND, we will first give an introduction to SAND and then briefly introduce SAND-2.

2.1 Specification of SAND block cipher

SAND is an AND-RX block cipher with Feistel construction, which has two versions SAND-64 and SAND-128. As SAND-2 only has 64-bit version, we only introduce SAND-64 here, it has 48 total rounds and 128-bit keysize.

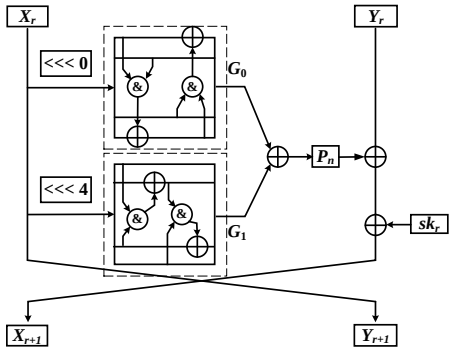

Fig. 1: Round Function of SAND-64.

Round function of SAND. As shown in Fig. 1, the left branch X_r firstly has a double expanding process and rotates with rotation constants (s_0, s_1), where $(s_0, s_1) = (0, 4)$ for its *Synthetic S-box* (SSb) equivalent representation. Non-linear components G_0 and G_1 are then applied parallelly. Before applying a

nibble equivalent permutation P_n and adding to the right branch Y_r, the outputs of G_0 and G_1 are compressed by XOR operations. For more details, we refer the reader to SAND design paper [CFS+22].

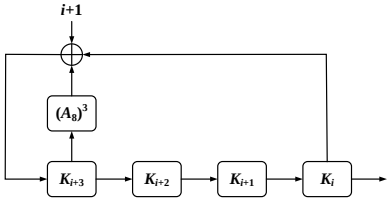

Fig. 2: Key schedule of SAND-64.

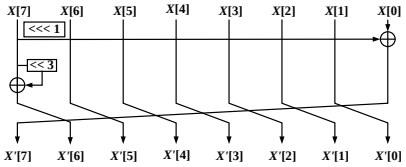

Fig. 3: Operation A_8 of SAND-64.

Key schedule. The 128-bit master key K can be viewed as four 32-bit words, *i.e.*, $K = K_3||K_2||K_1||K_0$. The update function of key schedule is shown in Fig. 2, and K_{i+4} can be calculated as below

$$K_{i+4} \leftarrow (A_8)^3(K_{i+3}) \oplus K_i \oplus (i+1),\ 0 \le i \le 43,$$

where $(A_8)^3$ denotes that the operation A_8 is applied to K_{i+3} for three times iteratively. A_8 is a nibble-based function and depicted in Fig. 3 where $X[j]$ denotes the j-th ($0 \le j < 8$) nibble of input X, *i.e.*, K_r($0 \le r \le 47$) can be divided into eight nibbles. Finally, the r-th round subkey sk_r will be loaded from K_r and added to the encryption state.

2.2 Specification of SAND-2 block cipher

SAND-2 adopts the same key schedule and the similar round function of SAND, it also changes the total rounds from 48 to 47.

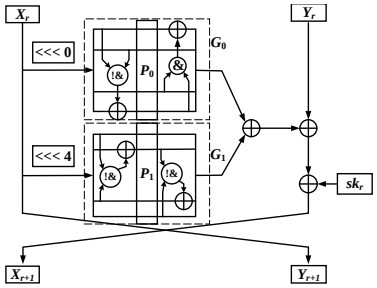

Fig. 4: Round function for even round.

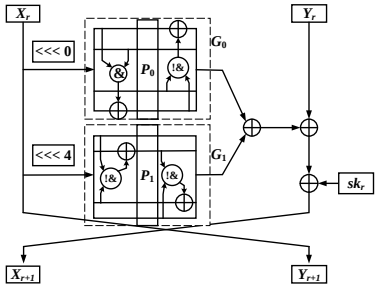

Fig. 5: Round function for odd round.

For the round function of SAND-2, as shown in Fig. 4, it firstly replaces some AND operations to NAND operations in G_0 and G_1. Then it uses two different bit permutations P_0 and P_1, as shown in Table 2, which are similar to the bit-based permutation adopted in ANT [CFF+19] block cipher. Finally, SAND-2 alternatively swaps NAND and AND operations in two layers of G_0 for even round and odd round (see in Fig. 5). For more details of SAND-2, please refer to its design paper [CLGH23].

Table 2: Bit permutations P_0 and P_1.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P_0(i)	28	23	26	1	0	27	30	5	4	31	2	9	8	3	6	13
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P_0(i)	12	7	10	17	16	11	14	21	20	15	18	25	24	19	22	29
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P_1(i)	20	27	2	29	24	31	6	1	28	3	10	5	0	7	14	9
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P_1(i)	4	11	18	13	8	15	22	17	12	19	26	21	16	23	30	25

3 Iterative and Full Round Distinguishers of SAND-2

In this section, we show how to exploit dependency properties existing in the round function of SAND-2 to construct iterative linear and differential characteristics. Then, full round linear and differential distinguishers can be both mounted. To enhance the probability of distinguishers, we evaluate the clustering effect for these distinguishers, experiments are also performed for verifications.

3.1 Linear Distinguishers of SAND-2

We firstly present a two-round iterative linear characteristic of SAND-2, as shown in Fig. 6, this is obtained by carefully observing two bit permutations P_0/P_1 and the rotation, then we have the following property.

Property 1. For the input bit with index $i_0 = 4 \times t + 3$[2] $(0 \leq t < 8)$ of G_0 and the input bit with index $i_1 = 4 \times (t+1) + 3$ of G_1, these two bits are derived from the same bit in X_r with index $4 \times t + 3$ and have a linear relation with $(4 \times t + 1)$-th output bit of G_0 and G_1 respectively.

We now give an example of this property as follows, and it should be noted that it has *other seven equivalent cases* for different choices of t. Also, this property is independent of the odd or even round.

[2] For simplicity, all bit indices are taken modulo 32 in the rest of the paper.

Example 1. For $i_0 = 31$, $P_0(i_0) = 29$, and $P_1(i_1) = 29$ when $i_1 = 3$. Due to the rotation $X_r \lll 0$ before G_0 and $X_r \lll 4$ before G_1, then the bit with index $i_0 = 31$ in G_0 and the bit with index $i_1 = 3$ in G_1 are derived from the same bit in X_r, which are marked as yellow in Fig. 6. Coincidentally, both these bits are linear related to the 29-th bit of the outputs of G_0 and G_1 respectively.

With Property 1, the two-round linear characteristic depicted in Fig. 6 with linear bias 2^{-2} (equivalent to linear probability 2^{-2}) is now constructed as below

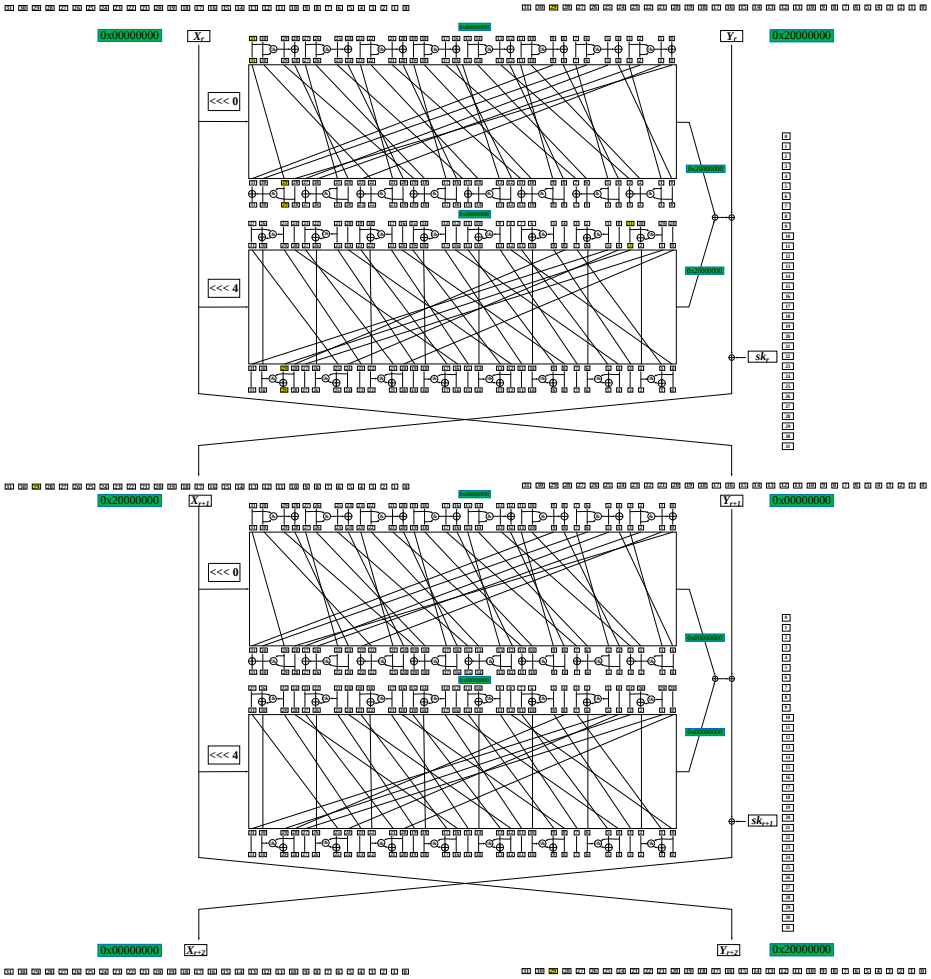

Fig. 6: Two-round iterative linear characteristic (active bit is marked as yellow).

1. Let only the 29-th bit mask of the right branch of the input be active, *i.e.,* $\Gamma Y_r[29] = 1$. All the rest bits of ΓY_r and ΓX_r are set to be zero mask. So, the input mask is $(\Gamma X_r, \Gamma Y_r) = (0x00000000, 0x20000000)$;
2. According to the propagation rule of linear mask, the output masks of G_0 and G_1 of the first round function are both with $0x20000000$;
3. In order to make the trail iterative, we let the mask $\Gamma Y_{r+1} = 0x00000000$. Then according to Property 1, the input masks of G_0 and G_1 of the first round function can be set to be $0x80000000$ and $0x00000008$ respectively;
4. With $\Gamma X_{r+1} = 0x20000000$, it leads to the output mask of the second round $(\Gamma X_{r+2}, \Gamma Y_{r+2}) = (\Gamma X_r, \Gamma Y_r) = (0x00000000, 0x20000000)$.

For the linear bias of this two-round trail, only the step 3 mentioned above produces the probability, that is, the second layer of G_1 of the first round function, as the 29-th output bit mask is non-zero, then the output mask of the corresponding NAND operation is also non-zero, which makes this NAND operation active with linear bias 2^{-2}. Thus, this two-round iterative linear characteristic[3] $(0x00000000, 0x20000000) \rightarrow (0x00000000, 0x20000000)$ has the linear bias with 2^{-2} and can start from both even or odd round number.

Longer trails, the clustering effect and experiments. With the presented two-round iterative trail above, longer number of rounds linear distinguishers can be naturally mounted. Especially, a *47-round (full round)* linear characteristic of SAND-2 can be obtained by iterating the two-round trail 23 times and appending one free round at the beginning, which has the linear bias with 2^{-24}. Not only does this trail invalidates the security of SAND-2 against linear attacks, but also its bias is even high enough to be a *practical full round distinguisher* with the complexity 2^{46}.

To verify these iterative trails, we firstly cluster the trails of different numbers of iterative times by the mature SAT/SMT automatic search method [AK18]. The results show that this kind of two-round iterative based linear trails has no significant clustering effect. Then, we perform experiments to evaluate the linear bias of these trails, which are given in Table 3 and match the results of clustering. In next section, we will use the 41-round linear distinguisher with the bias 2^{-21} to mount full round key recovery attack on SAND-2.

3.2 Differential Distinguishers of SAND-2

Similar to finding linear distinguishers of SAND-2, we now show how to construct iterative differential trails of SAND-2 and then try to approximate differential probability by considering the clustering effect.

We firstly divide the input bits of G_0 (G_1) by its 4-bit output, as shown in Table 4 and Table 5, where the six bit indices of each row is derived from the

[3] Note that this trail also has other seven equivalent cases, all these can be used to mount longer distinguishers and key recovery attacks. In the rest of the paper, we will only focus on evaluating one case, but the other seven cases will be similar.

Table 3: Experiments of the iterative linear characteristics.

Round	Theoretical linear bias	Experimental linear bias	Test data
2	2^{-2}	$2^{-2.00}$	2^{26}
4	2^{-3}	$2^{-3.00}$	2^{26}
6	2^{-4}	$2^{-4.00}$	2^{26}
8	2^{-5}	$2^{-4.99}$	2^{26}
16	2^{-9}	$2^{-9.02}$	2^{26}
24	2^{-13}	$2^{-12.91}$	2^{26}
26	2^{-14}	$2^{-13.89}$	2^{28}

left branch state X_r and $G_0[3-0]$ represents the lowest nibble of the output of G_0 (similarly for G_1). Then, still by carefully observing the inputs of G_0 and G_1, it has the following property.

Property 2. When all input bit differences of X_r are zero, except the $(4\times t+3)$-th $(0 \leq t < 8)$ bit, that is $\Delta X_r[4 \times t + 3] = 1$, it then has

- For the $(t-1)$-th[4] nibble of G_0, its input difference is $0b001000$ (binary representation). Then, its possible 4-bit output differences are $0b0000$ with probability $\frac{1}{2}$, $0b0001$ with probability $\frac{1}{4}$ or $0b1001$ with probability $\frac{1}{4}$;
- For the t-th nibble of G_0, its input difference is $0b000001$. Then, its possible 4-bit output differences are $0b0010$ with probability $\frac{1}{2}$ or $0b1010$ with probability $\frac{1}{2}$;
- For the $(t-7)$-th nibble of G_1, its input difference is $0b000100$. Then, its possible 4-bit output differences are $0b0000$ with probability $\frac{1}{2}$, $0b0110$ with probability $\frac{1}{4}$ or $0b0100$ with probability $\frac{1}{4}$;
- For the t-th nibble of G_1, its input difference is $0b001000$, its 4-bit output difference must be $0b0010$.

Table 4: Grouping input bit index of G_0.

Nibbles	Bit index of X_r					
$G_0[3-0]$	13	10	7	6	4	3
$G_0[7-4]$	17	14	11	10	8	7
$G_0[11-8]$	21	18	15	14	12	11
$G_0[15-12]$	25	22	19	18	16	15
$G_0[19-16]$	29	26	23	22	20	19
$G_0[23-20]$	1	30	27	26	24	23
$G_0[27-24]$	5	2	31	30	28	27
$G_0[31-28]$	9	6	3	2	0	31

Table 5: Grouping input bit index of G_1.

Nibbles	Bit index of X_r					
$G_1[3-0]$	8	5	3	31	30	29
$G_1[7-4]$	12	9	7	3	2	1
$G_1[11-8]$	16	13	11	7	6	5
$G_1[15-12]$	20	17	15	11	10	9
$G_1[19-16]$	24	21	19	15	14	13
$G_1[23-20]$	28	25	23	19	18	17
$G_1[27-24]$	0	29	27	23	22	21
$G_1[31-28]$	4	1	31	27	26	25

[4] For simplicity, the index number of the nibble takes modulo 8.

With Property 2, we can easily construct an iterative two-round differential characteristic with probability 2^{-3} as below

1. Let only the 31-th bit difference ($t = 7$) of the left branch of the input be active, *i.e.,* $\Delta X_r[31] = 1$. All the rest bits of ΔX_r and ΔY_r are set to be zero difference. So, the input difference is $(\Delta X_r, \Delta Y_r) = (0x80000000, 0x00000000)$;
2. In order to make this trail iterative, according to Property 2, we let the 4-bit output differences of the 6-th nibble of G_0 and the 0-th nibble of G_1 be both zero, and 4-bit output difference of the 7-th nibbles of G_0 be $0b0010$, which then can cancel the difference of the corresponding nibble of G_1;
3. With $\Delta X_{r+1} = 0x00000000$, it leads to the output difference of the second round $(\Delta X_{r+2}, \Delta Y_{r+2}) = (\Delta X_r, \Delta Y_r) = (0x80000000, 0x00000000)$.

For the differential probability of this two-round trail, only the step 2 mentioned above produces the probability 2^{-3}, that is, cancelling all differences at the compression. Thus, this two-round differential trail[5] $(0x80000000, 0x00000000) \rightarrow (0x80000000, 0x00000000)$ has the differential probability 2^{-3} and can start from both even or odd round number.

Approximation of differential probabilities. In order to obtain a more accurate differential probability of the iterative based distinguishers for later attacks, we try to approximate the probability of these iterative distinguishers, however, the method we proposed in the following still cannot capture all trails for the given differential, but it can effectively cluster the differential with high probability, which usually dominates the final probability of a differential. We then also use SAT/SMT automatic search method to evaluate the clustering effect, which shows our method is efficient and effective to approximate the probability of such iterative differentials of SAND-2. The following property is firstly introduced, which can be partly derived from Property 2.

Property 3. For one round of SAND-2, it has the following differential characteristics and corresponding probability P:

- $(0x80000000, 0x00000000) \xrightarrow{1-round} (0x00000000, 0x80000000)$, $P = 2^{-3}$;
- $(0x80000000, 0x00000000) \xrightarrow{1-round} (0x80000000, 0x80000000)$, $P = 2^{-3}$;
- $(0x80000000, 0x80000000) \xrightarrow{1-round} (0x00000000, 0x80000000)$, $P = 2^{-3}$;
- $(0x80000000, 0x80000000) \xrightarrow{1-round} (0x80000000, 0x80000000)$, $P = 2^{-3}$;
- $(0x00000000, 0x80000000) \xrightarrow{1-round} (0x80000000, 0x00000000)$, $P = 1$.

We note that the case in Property 3 is also just one of the eight equivalent cases. Based on Property 3, except the above presented two-round iterative trail with probability 2^{-3}, we can also construct an $(m + 3)$-round differential trail with probability $2^{-3(m+2)}$ as below:

$$(0x8, 0x0) \rightarrow (0x8, 0x8) \xrightarrow{m-round} (0x8, 0x8) \rightarrow (0x0, 0x8) \rightarrow (0x8, 0x0).$$

[5] Similarly, this trail also has other seven equivalent cases, which can be both used to mount longer distinguishers and key recovery attacks.

With this configurable trail, it may bring us many different characteristics that can be clustered to enhance the final probability. Thus, we provide the following formulas to do such clustering process.

Proposition 1. *For a given even number of rounds $N_r = 2n_r$ ($n_r \in \mathbf{Z}^+$), for a fixed probability $P = 2^{-p}$ ($p \in \mathbf{Z}^+$), it has*

$$2j + \sum_{i=0, m_i=i}^{min(N_r-3, \lfloor \frac{p}{3}-2 \rfloor)} (m_i+3)k_i = N_r,$$

$$3j + \sum_{i=0, m_i=i}^{min(N_r-3, \lfloor \frac{p}{3}-2 \rfloor)} 3(m_i+2)k_i = p,$$

where j ($0 \leq j \leq n_r$) denotes the iterative times of the two-round iterative differential characteristic and k_i ($0 \leq k_i \leq \lfloor \frac{p}{3(m_i+2)} \rfloor$) denotes the number of (m_i+3)-round iterative differential. We can obtain a set of values for $(j, k_0, k_1, ...)$ which denotes the number of different short-round iterative differential characteristic. Then we iterate the position of these short-round iterative differential characteristic, equivalent to calculating a permutation combination number, which corresponds to different trails that can be clustered.

In order to check whether the method presented above can effectively approximate the probability of longer rounds iterative distinguishers, we also apply the SAT/SMT based search method to do the clustering for 8-round and 16-round iterative differential trails with probability greater than 2^{-30} and match the results clustered by Proposition 1. Then the experiments to compare the theoretical and experimental probabilities are also performed for several distinguishers, which shows the effectiveness of our proposed method and are given in Table 6. It can be observed that these iterative differential trails have slight clustering effect.

Table 6: Experiments of the iterative differentials of SAND-2.

Round	Theoretical probability	Experimental probability	Test data
2	$2^{-3.00}$	$2^{-3.00}$	2^{27}
4	$2^{-5.83}$	$2^{-5.81}$	2^{27}
8	$2^{-11.13}$	$2^{-11.08}$	2^{27}
16	$2^{-21.21}$	$2^{-21.02}$	2^{27}

Distinguishers for full round attacks. With the effective and efficient evaluation method presented above, we then mount 40-round, 42-round and 46-round iterative distinguishers. It should be noted that for such longer rounds, SAT/SMT based method is already very inefficient due to the size of models thus cannot provide tight bounds of the final probability. For the formulas in Proposition 1, we also limit and select part m_i for the calculation considering the efficiency, thus

just providing a lower bound of the probability. However, it is still high enough to launch full round attacks and invalidate security bounds given in SAND-2 design paper [CLGH23, Table 17].

- For 46-round distinguisher, it has the probability $2^{-58.60}$. When one round is added to its head with probability 1, this can lead to a *full round differential distinguisher* of SAND-2.
- In order to launch *full round key recovery attacks* on SAND-2, we mount a 43-round (extended from 42-round) and a 41-round (extended from 40-round) distinguisher with probability $2^{-53.62}$ and $2^{-51.13}$ respectively.

Remark: In this section, we construct iterative differential and linear distinguishers, which already lead to full round distinguish attacks. Some distinguishers will be later used to mount full round key recovery attacks, and we summarize these distinguishers in Table 7. It should be noted that we only perform experiments on some short rounds distinguishers of SAND-2 due to our inefficient software implementation[6] of SAND-2 and the limited computing resources.

Table 7: Summary of the differential and linear distinguishers of SAND-2.

Type	Distinguisher	Probability	Usage
Linear	$(0x2, 0x0) \xrightarrow{47\text{-}round} (0x0, 0x2)$	2^{-46}	Distinguishing attack
Linear	$(0x2, 0x0) \xrightarrow{41\text{-}round} (0x0, 0x2)$	2^{-40}	Key recovery attack
Differential	$(0x0, 0x8) \xrightarrow{47\text{-}round} (0x8, 0x0)$	$2^{-58.60}$	Distinguishing attack
Differential	$(0x0, 0x8) \xrightarrow{43\text{-}round} (0x8, 0x0)$	$2^{-53.62}$	Key recovery attack
Differential	$(0x0, 0x8) \xrightarrow{41\text{-}round} (0x8, 0x0)$	$2^{-51.13}$	Key recovery attack

4 Key Recovery Attacks on SAND-2

In this section, we give the *full round key recovery attacks* based on the 41-round linear distinguisher and 43-round differential distinguisher presented above. For linear attack, the time, data and memory complexities are $2^{48.23}$ full round encryptions, $2^{45.50}$ known-plaintexts and $2^{35.00}$ respectively. For differential attack, the time, data and memory complexities are $2^{64.10}$ full round encryptions, $2^{60.20}$ chosen-plaintexts and $2^{57.20}$ respectively.

It should be noted that a full round key recovery attack based on 41-round differential distinguisher is also mounted, which has a lower data complexity $2^{53.13}$ but a higher time complexity $2^{73.70}$. We provide a detailed description about this result in Appendix A.

[6] Because SAND-2 adopts two different bit permutation layers P_0/P_1 and different round functions in even or odd round.

4.1 Full Round Linear Attack

In the attack, we both append three rounds before and after the 41-round linear distinguisher. The key recovery attack is illustrated in Fig. 7 and Fig. 8, where X_i and Y_i denote the 32-bit input to the left and right branches, $G_0(X_i)$ and $G_1(X_i)$ represent the 32-bit output of function G_0 and G_1 in the i-th round, W_i records the XOR value of output of G_0, G_1 and Y_i in the head(resp. Y_{i+1} in the tail), rk_i stands for the i-th round key and M_i is the XOR value of W_i and rk_i. In the following, we use $X_i[j]$ to represent the j-th bit of X_i and the least significant bit is $X_i[0]$. And the white cell denotes the linear mask of the bit is zero, the yellow cell represents the linear mask of the bit is non-zero, the blue cell denotes the value of the bit should be computed, the red cell represents the subkey bits that are involved in the partial encryption and decryption phases and the sequence numbers represent the order in which the key bits are guessed.

Suppose that the number of required plaintext-ciphertext pairs is N_L. The attack is realised with the following steps.

1. Guess 4-bit subkey value $rk_0[8, 10\text{-}12]$ and allocate a counter $C_1^L[z_1]$ for each of 2^{35} possible values of

$$\begin{aligned} z_1 =& X_{46}[4, 25\text{-}27]||W_0[0, 2, 5, 21\text{-}23, 27\text{-}31]||W_1[4, 25\text{-}27]|| \\ & W_{46}[0, 2, 5, 8, 10\text{-}12, 21\text{-}23, 27\text{-}31]||t_0, \end{aligned}$$

   where $t_0 = G_0(X_0)[29] \oplus G_1(X_0)[29] \oplus Y_0[29] \oplus G_0(X_{46})[29] \oplus G_1(X_{46})[29] \oplus Y_{47}[29]$. Then, for each possible 4-bit subkey value $rk_0[8, 10\text{-}12]$, we compute the value of z_1 and update $C_1^L[z_1]$ with $C_1^L[z_1] + 1$, thus the dominant time complexity is $N_L \cdot 2^4$ memory accesses to a table with 2^{35} elements.
2. Guess 6-bit subkey value $rk_0[0, 2, 21\text{-}23, 27]$ and allocate a counter $C_2^L[z_2]$ for each of 2^{30} possible values of

$$\begin{aligned} z_2 =& X_{46}[4, 25\text{-}27]||W_0[5, 28\text{-}31]||M_0[27]||W_1[4, 25\text{-}27]|| \\ & W_{46}[0, 2, 5, 8, 10\text{-}12, 21\text{-}23, 27\text{-}31]||t_1, \end{aligned}$$

   where $t_1 = t_0$ and $M_0[27]$ records the value of $rk_0[27]$. For each possible 6-bit subkey value $rk_0[0, 2, 21\text{-}23, 27]$, we compute the value of z_2 and update $C_2^L[z_2]$ with $C_2^L[z_2] + C_1^L[z_1]$, thus the dominant time complexity of this step is $2^{35} \cdot 2^4 \cdot 2^6 = 2^{45}$ memory accesses to a table with 2^{30} elements.
3. Guess 5-bit subkey value $rk_0[5, 28\text{-}31]$ and allocate a counter $C_3^L[z_3]$ for each of 2^{24} possible values of

$$z_3 = X_{46}[4, 25\text{-}27]||W_1[4, 25\text{-}27]||W_{46}[0, 2, 5, 8, 10\text{-}12, 21\text{-}23, 27\text{-}31]||t_2,$$

   where $t_2 = t_1$. For each possible 5-bit subkey value $rk_0[5, 28\text{-}31]$, we compute the value of z_3 and update $C_3^L[z_3]$ with $C_3^L[z_3] + C_2^L[z_2]$, thus the dominant time complexity of this step is $2^{30} \cdot 2^{10} \cdot 2^5 = 2^{45}$ memory accesses to a table with 2^{24} elements.

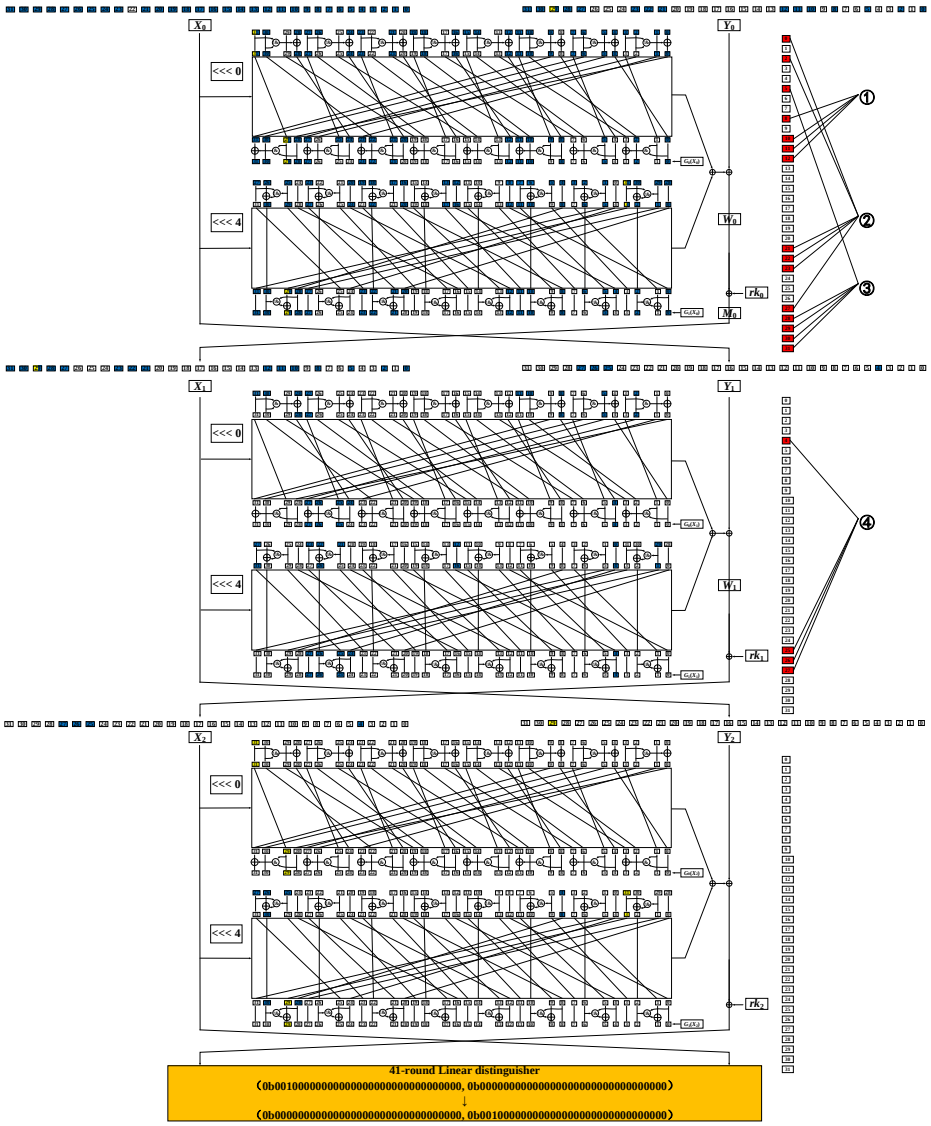

Fig. 7: The head of Linear key recovery attack on full round SAND-2.

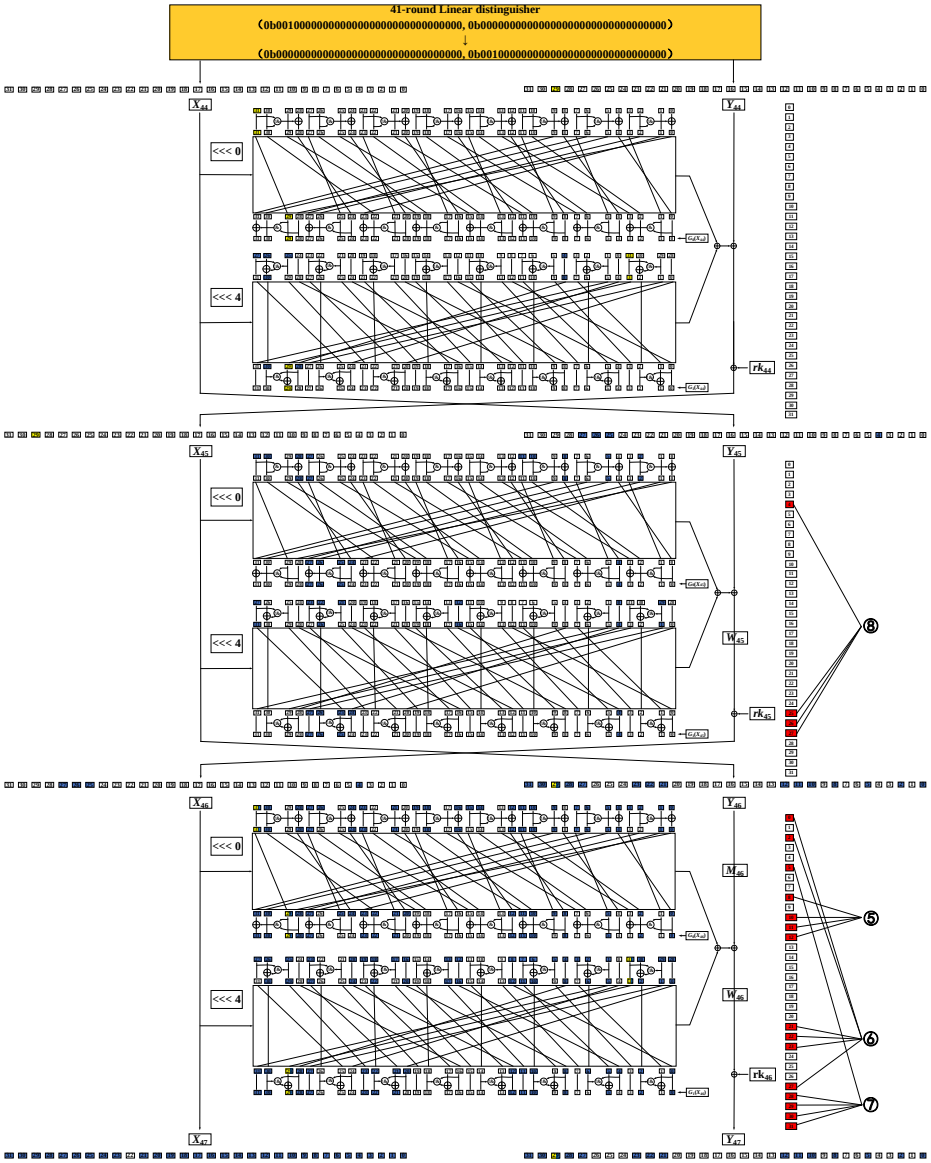

Fig. 8: The tail of Linear key recovery attack on full round SAND-2.

4. Guess 4-bit subkey value $rk_1[4, 25\text{-}27]$ and allocate a counter $C_4^L[z_4]$ for each of 2^{20} possible values of

$$z_4 = X_{46}[4, 25\text{-}27]||W_{46}[0, 2, 5, 8, 10\text{-}12, 21\text{-}23, 27\text{-}31]||t_3,$$

where $t_3 = t_2 \oplus G_1(X_2)[29]$. For each possible 4-bit subkey value $rk_1[4, 25\text{-}27]$, we compute the value of z_3 and update $C_4^L[z_4]$ with $C_4^L[z_4] + C_3^L[z_3]$. The dominant time complexity of this step is $2^{24} \cdot 2^{15} \cdot 2^4 = 2^{43}$ memory accesses to a table with 2^{20} elements.
5. Guess 4-bit subkey value $rk_{46}[8, 10\text{-}12]$ and allocate a counter $C_5^L[z_5]$ for each of 2^{16} possible values of

$$z_5 = W_{46}[0, 2, 5, 21\text{-}23, 27\text{-}31]||W_{45}[4, 25\text{-}27]||t_4,$$

where $t_4 = t_3$. For each possible 4-bit subkey value $rk_{46}[8, 10\text{-}12]$, we compute the value of z_5 and update $C_5^L[z_5]$ with $C_5^L[z_5] + C_4^L[z_4]$. The dominant time complexity of this step is $2^{20} \cdot 2^{19} \cdot 2^4 = 2^{43}$ memory accesses to a table with 2^{16} elements.
6. Guess 6-bit subkey value $rk_{46}[0, 2, 21\text{-}23, 27]$ and allocate a counter $C_6^L[z_6]$ for each of 2^{11} possible values of

$$z_6 = W_{46}[5, 28\text{-}31]||M_{46}[27]||W_{45}[4, 25\text{-}27]||t_5,$$

where $t_5 = t_4$. For each possible 6-bit subkey value $rk_{46}[0, 2, 21\text{-}23, 27]$, we compute the value of z_6 and update $C_6^L[z_6]$ with $C_6^L[z_6] + C_5^L[z_5]$. The dominant time complexity of this step is $2^{16} \cdot 2^{23} \cdot 2^6 = 2^{45}$ memory accesses to a table with 2^{11} elements.
7. Guess 5-bit subkey value $rk_{46}[5, 28\text{-}31]$ and allocate a counter $C_7^L[z_7]$ for each of 2^5 possible values of

$$z_7 = W_{45}[4, 25\text{-}27]||t_6,$$

where $t_6 = t_5$. For each possible 5-bit subkey value $rk_{46}[5, 28\text{-}31]$, we compute the value of z_7 and update $C_7^L[z_7]$ with $C_7^L[z_7] + C_6^L[z_6]$. The number of memory accesses in this step is $2^{11} \cdot 2^{29} \cdot 2^5 = 2^{45}$.
8. Guess 4-bit subkey value $rk_{45}[4, 25\text{-}27]$ and initialize a counter $Counter_L$. Then compute the value of $t_7 = t_6 \oplus G_1(X_{44})[29]$ for each possible 4-bit subkey value $rk_{45}[4, 25\text{-}27]$. If $t_7 = 0$, we update $Counter_L$ with $Counter_L + C_7^L[z_7]$. The number of memory accesses in this step is $2^5 \cdot 2^{34} \cdot 2^4 = 2^{43}$.
9. The key guess will be accepted as a candidate if the counter $Counter_L$ satisfies $|Counter_L/N_L - 0.5| > \tau_L$, where τ_L is the threshold used in [SWW21].
10. As mentioned above, the mask "2" of linear distinguisher can be placed in the remaining 7 position. Thus we can use three of them to recover key. Each distinguisher involves 38-bit key and can recover 84 bits when considering them together (30 bits are overlapped), which are shown in the Table 8.
11. Then do exhaustive search for all keys that correspond to the guessed 84-bit subkey bits against a maximum of two plaintext-ciphertext pairs.

Table 8: Key bits involved in the 41-round distinguishers.

Distinguisher	Key bits involved in the distinguisher	All 84-bit key
(0x20000000,0x00000000) ↓ (0x00000000,0x20000000)	rk_0[0, 2, 5, 8, 10-12, 21-23, 27-31] rk_{46}[0, 2, 5, 8, 10-12, 21-23, 27-31] rk_1[4, 25-27] rk_{45}[4, 25-27]	rk_0 [0-8, 10-15, 17-31] rk_{46} [0-8, 10-15, 17-31] rk_1[0, 4, 17-19 21-23, 25-28] rk_{45}[0, 4, 17-19 21-23, 25-28]
(0x02000000,0x00000000) ↓ (0x00000000,0x02000000)	rk_0[1, 4, 6-8, 17-19, 23-28, 30] rk_{46}[1, 4, 6-8, 17-19, 23-28, 30] rk_1[0, 21-23] rk_{45}[0, 21-23]	
(0x00200000,0x00000000) ↓ (0x00000000,0x00200000)	rk_0[0, 2-4, 13-15, 19-24, 26, 29] rk_{46}[0, 2-4, 13-15, 19-24, 26, 29] rk_1[17-19, 28] rk_{45}[17-19, 28]	

Complexity Analysis. As we leave 2^{16} candidates, that means the advantage [Sel08] of the attack as a=22. For three distinguishers, there are a total of $2^{16} \cdot 2^{16} \cdot 2^{16}$ candidate keys remaining. Then we set the number of pairs N_L as $2^{45.5}$, so the data complexity of this attack is $2^{45.5}$. And according to [SN14], we consider one memory access as a half round encryption. So, the time complexity of this attack can be computed as follows.

$$3 \times (N_L \cdot 2^4 + 2^{45} \times 4 + 2^{43} \times 3) \times \frac{1}{2} \times \frac{1}{47} + ((2^{16})^3 + 2^{128-84}) \times (1 + 2^{-64}),$$

where $((2^{16})^3 + 2^{128-84}) \times (1 + 2^{-64})$ denotes the time complexity of step 11. Then, the time complexity of the attack is about $2^{48.23}$ full round encryptions. $C_1^L[z_1]$ dominates the memory complexity which is roughly 2^{35}. We calculate the success probability by the following formula in [BN17]:

$$P_s \approx \Phi\left(\frac{c \cdot \sqrt{N_L} - \Phi^{-1}(1 - 2^{-(a+1)}) \cdot \sqrt{1 + N_L \cdot 2^{-n}}}{\sqrt{1 + N_L \cdot (ELP - c^2)}}\right),$$

where $\Phi(\cdot)$ is the normal distribution and n is block size. The variable c denotes the approximation of the absolute value of the correlation related to the dominant linear characteristic and the expected linear potential, denoted as ELP, of the approximation is calculated as the sum of squared correlations across all characteristics associated with it. In our attack, $ELP = c^2$, thus the success probability for one such attack is $P_s = 94.07\%$ and $(94.07\%)^3 = 83.24\%$.

4.2 Full Round Differential Attack

In the attack, we both append two rounds before and after the 43-round distinguisher. The key recovery attack is illustrated in Fig. 9, where the white cell denotes the difference of the bit is zero, the yellow cell represents the difference

of the bit is non-zero, the blue cell denotes the difference of the bit can be zero or non-zero and the red cell represents the value of the bit needs to be computed for the intermediate states and for the round key that denotes being guessed.

Data collection. We can construct structures at the position of (X_0, Y_0). In each structure, the 43 bits

$$X_0[0, 3\text{-}23, 25\text{-}26, 28\text{-}30]||Y_0[0, 3\text{-}4, 7\text{-}15, 18\text{-}19, 21\text{-}22]$$

with the difference being zero in Fig. 9 are fixed, and the value of the remaning 21 bits are traversed. Thus, 2^{41} pairs can be generated with one structure composed of 2^{21} plaintexts.

Table 9: Conditions for key recovery and filter probabilities on SAND-2.

Condition	Filter	Probability
(C1)	$\Delta X_{47}[0, 3\text{-}23, 25\text{-}26, 28\text{-}30]\|\|\Delta Y_{47}[0, 3\text{-}4, 7\text{-}15, 18\text{-}19, 21\text{-}22] = 0$	2^{-43}
(C2)	$\Delta X_1[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30] = 0, \Delta X_1[31] = 1$	2^{-16}
(C3)	$\Delta Y_{46}[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30] = 0, \Delta Y_{46}[31] = 1$	2^{-16}
(C4)	$\Delta X_2[1\text{-}2, 24, 27, 31] = 0$	2^{-5}
(C5)	$\Delta Y_{45}[1\text{-}2, 24, 27, 31] = 0$	2^{-5}

Key recovery. In the attack, we prepare N_s structures and obtain $N_1 = N_s \cdot 2^{41}$ pairs. Thus, the data complexity of the attack is $N_s \cdot 2^{21}$. The detailed attack is realised with the following steps and we list all filter conditions in Table 9.

1. For each pair $P = (X_0, Y_0)$ and $P' = (X'_0, Y'_0)$, we obtain the corresponding values of the ciphertexts $C = (X_{47}, Y_{47})$ and $C' = (X'_{47}, Y'_{47})$ by querying the oracle. The time complexity of this step is $N_s \cdot 2^{21}$ full round encryptions.
2. Denoising over ciphertexts: the 43 bits of ciphertexts with the difference being zero and check the condition (C1), $N_1 \cdot 2^{-43}$ pairs will be left;
3. For each pair $P = (X_0, Y_0)$ and $P' = (X'_0, Y'_0)$, we first calculate $\Delta X_1 = X_1 \oplus X'_1$ without guessing any key bits and check the condition (C2), then $N_1 \cdot 2^{-43} \cdot 2^{-16}$ pairs will be left. This step involves 18 S-box operation, thus the time complexity is $2 \cdot N_1 \cdot 2^{-43} \times 18 \times 1/32 \times 1/47$ full round encryptions.
4. For each pair $C = (X_{47}, Y_{47})$ and $C' = (X'_{47}, Y'_{47})$, we can calculate ΔY_{46} without guessing key bits and check the condition (C3), then $N_1 \cdot 2^{-43} \cdot 2^{-16} \cdot 2^{-16}$ pairs will be left. Similarly, this step involves 18 S-box operation, thus the time complexity is $2 \cdot N_1 \cdot 2^{-43} \cdot 2^{-16} \times 18 \times 1/32 \times 1/47$ full round encryptions.
5. Guess 7 bits of rk_0. We can compute $X_2[1\text{-}2, 24, 27, 31]$ and $X'_2[1\text{-}2, 24, 27, 31]$ for each possible 7-bit subkey value $rk_0[0, 2\text{-}3, 8, 27, 29\text{-}30]$ and check the condition (C4), then $N_1 \cdot 2^{-43} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^{-5}$ pairs will be left. This step involves 6 S-box operation, thus the time complexity is $2 \cdot N_1 \cdot 2^{-43} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^7 \times 6 \times 1/32 \times 1/47$ full round encryptions.

6. Guess 7 bits of rk_{46}. We can compute ΔY_{45}[1-2, 24, 27, 31] for each possible 7-bit subkey value rk_{46}[0, 2-3, 8, 27, 29-30] and check the condition (C5), then $N_1 \cdot 2^{-43} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^{-5} \cdot 2^{-5}$ pairs will be left. Similarly, the time complexity is $2 \cdot N_1 \cdot 2^{-43} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^{-5} \cdot 2^7 \times 6 \times 1/32 \times 1/47$ full round encryptions.
7. As mentioned above, the difference "8" of Differential distinguisher can be placed in the remaining 7 position. Thus we can use all of them to recover key. Each distinguisher involves the 14-bit key, and the all can recover a total of 64 bits which cover exactly rk_0 and rk_{46}.
8. Then do exhaustive search for all keys which correspond to the guessed 64-bit subkey bits against a maximum of two plaintext-ciphertext pairs.

Complexity Analysis. We set a counter to record the number of right pairs that validate the input and output differences of the 43-round distinguisher. With the analysis above, for random key guesses, the number of right pairs is about $N_1 \cdot 2^{-85}$. For the right key guess, the number of right pairs is expected to be $N_1 \cdot 2^{-21} \cdot 2^{-53.62}$, where 2^{-21} is the probability of the difference of plaintext to the head of the distinguisher and $2^{-53.62}$ is the probability of the distinguisher. In order to get higher success probability we set the number of right pair μ is 6 and the signal-to-noise ratio $S_N = \frac{N_1 \cdot 2^{-21} \cdot 2^{-53.62}}{N_1 \cdot 2^{-85}} = 2^{10.38}$. So the pairs N_1 is $2^{77.20}$ and corresponding N_s is $2^{36.20}$. For eight distinguishers, the data requirement of the attack is $8 \times 2^{36.20} \cdot 2^{21} = 2^{60.20}$ chosen plaintexts. As we leave only one best candidate, that means advantage a is 14. So the time complexity of this attack can be computed as follows.

$$8 \times (2^{36.20} \cdot 2^{21} + N_1 \cdot 2^{-42.97}) + (1 + 2^{64}) \times (1 + 2^{-64}),$$

where $(1+2^{64}) \times (1+2^{-64})$ denotes the time complexity of step 8. Then, the time complexity of this attack is about $2^{64.10}$ full round encryptions. We calculate the success probability by the following formula in [Sel08]:

$$P_s = \Phi(\frac{\sqrt{\mu S_N} - \Phi^{-1}(1 - 2^{-a})}{\sqrt{S_N + 1}}),$$

thus the success probability for one such attack is $P_s = 99.04\%$ and $(99.04\%)^8 = 92.61\%$ for the whole attack. Since we should record the right pairs, the memory complexity of this attack is roughly $2^{57.20}$.

5 Conclusion

In this paper, we present full round distinguishing and key recovery attacks on lightweight block cipher SAND-2 in single-key setting. Our attacks exploit iterative distinguishers with high probability (*e.g.,* the time complexities of linear distinguishing and key recovery attacks are both even lower than 2^{50}), which are derived from the dependencies of the round function of SAND-2. Moreover, we believe that our attacks provide the insight for designers about the importance of applying extensive and in-depth security analysis under designers' responsibility.

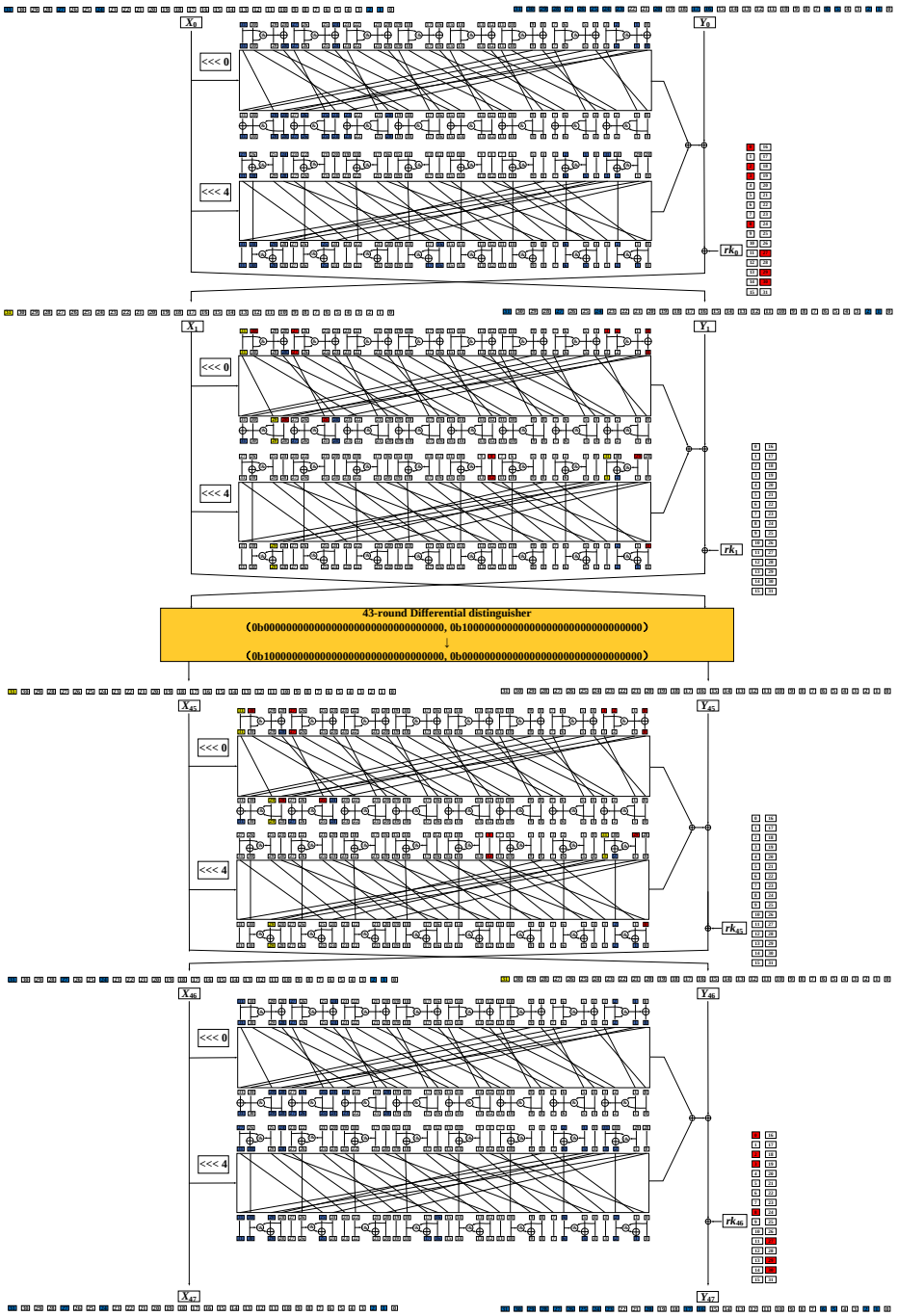

Fig. 9: Differential key recovery attack on full round SAND-2.

Acknowledgments. We sincerely thank the anonymous reviewers for providing valuable comments to help us improve the overall quality of the paper. This work is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702 & 2022YFB2701700), the National Natural Science Foundation of China (Grant No. 62032014), the Shandong Provincial Natural Science Foundation (Grant No. ZR2020MF053), the Major Basic Research Project of Natural Science Foundation of Shandong Province (Grant No. ZR202010220025), Department of Science & Technology of Shandong Province (Grant No. SYS202201), and Quan Cheng Laboratory (Grant No. QCLZD202306).

References

AK18. Ralph Ankele and Stefan Kölbl. Mind the gap - A closer look at the security of block ciphers against differential cryptanalysis. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 163–190. Springer, 2018.

Bih94. Eli Biham. New types of cryptanalytic attacks using related keys. *J. Cryptology*, 7(4):229–246, 1994.

BJK+16. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 123–153, 2016.

BN17. Céline Blondeau and Kaisa Nyberg. Joint data and key distribution of simple, multiple, and multidimensional linear cryptanalysis test statistic and its impact to data complexity. *Des. Codes Cryptogr.*, 82(1-2):319–349, 2017.

BPSP16. Gaurav Bansod, Abhijit Patil, Swapnil Sutar, and Narayan Pisharoty. ANU: an ultra lightweight cipher design for security in IoT. *Security and Communication Networks*, 9(18):5238–5251, 2016.

BS91. Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.

BSS+13. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.*, page 404, 2013.

CFF+19. Shiyao Chen, Yanhong Fan, Yong Fu, Luning Huang, and Meiqin Wang. On the design of ant family block ciphers. *Journal of Cryptologic Research.*, 6(6):748–759, 2019.

CFS+22. Shiyao Chen, Yanhong Fan, Ling Sun, Yong Fu, Haibo Zhou, Yongqing Li, Meiqin Wang, Weijia Wang, and Chun Guo. SAND: an AND-RX feistel lightweight block cipher supporting s-box-based security evaluations. *Des. Codes Cryptogr.*, 90(1):155–198, 2022.

CLGH23. Wen Chen, Lang Li, Ying Guo, and Ying Huang. SAND-2: an optimized implementation of lightweight block cipher. *Integr.*, 91:23–34, 2023.

Knu91. Lars R. Knudsen. Cryptanalysis of LOKI. In *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, pages 22–35, 1991.

Mat93. Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

Sas18. Yu Sasaki. Related-key boomerang attacks on full ANU lightweight block cipher. In *International Conference on Applied Cryptography and Network Security*, pages 421–439. Springer, 2018.

Sel08. Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *J. Cryptol.*, 21(1):131–147, 2008.

SN14. Hadi Soleimany and Kaisa Nyberg. Zero-correlation linear cryptanalysis of reduced-round lblock. *Des. Codes Cryptogr.*, 73(2):683–698, 2014.

SWW21. Ling Sun, Wei Wang, and Meiqin Wang. Improved attacks on GIFT-64. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, volume 13203 of *Lecture Notes in Computer Science*, pages 246–265. Springer, 2021.

A Full Round Differential Attack based on 41-round Distinguisher

In this attack, we both append three rounds before and after the 41-round distinguisher. The key recovery attack is illustrated in Fig. 10 and Fig. 11, where the meaning of different cells is the same as Fig. 9.

Data collection. We can construct structures at the position of (X_0, Y_0). In each structure, the 19 bits

$$X_0[0, 3\text{-}4, 7\text{-}15, 18\text{-}19, 21\text{-}22]||Y_0[4, 11, 14]$$

with the difference being zero in Fig. 10 are fixed, and the value of the remaning 45 bits are traversed. Thus, 2^{89} pairs can be generated with one structure composed of 2^{45} plaintexts.

Key recovery. In the attack, we prepare N_s structures and obtain $N_1 = N_s \cdot 2^{89}$ pairs. Thus, the data complexity of the attack is $N_s \cdot 2^{45}$. The detailed attack is realised with the following steps and we list all filter conditions in Table 10.

1. For each pair $P = (X_0, Y_0)$ and $P' = (X'_0, Y'_0)$, we obtain the corresponding values of the ciphertexts $C = (X_{47}, Y_{47})$ and $C' = (X'_{47}, Y'_{47})$ by querying the oracle. The time complexity of this step is $N_s \cdot 2^{45}$ full round encryptions.
2. Denoising over ciphertexts: the 19 bits of ciphertexts with the difference being zero and check the condition (C1), $N_1 \cdot 2^{-19}$ pairs will be left;

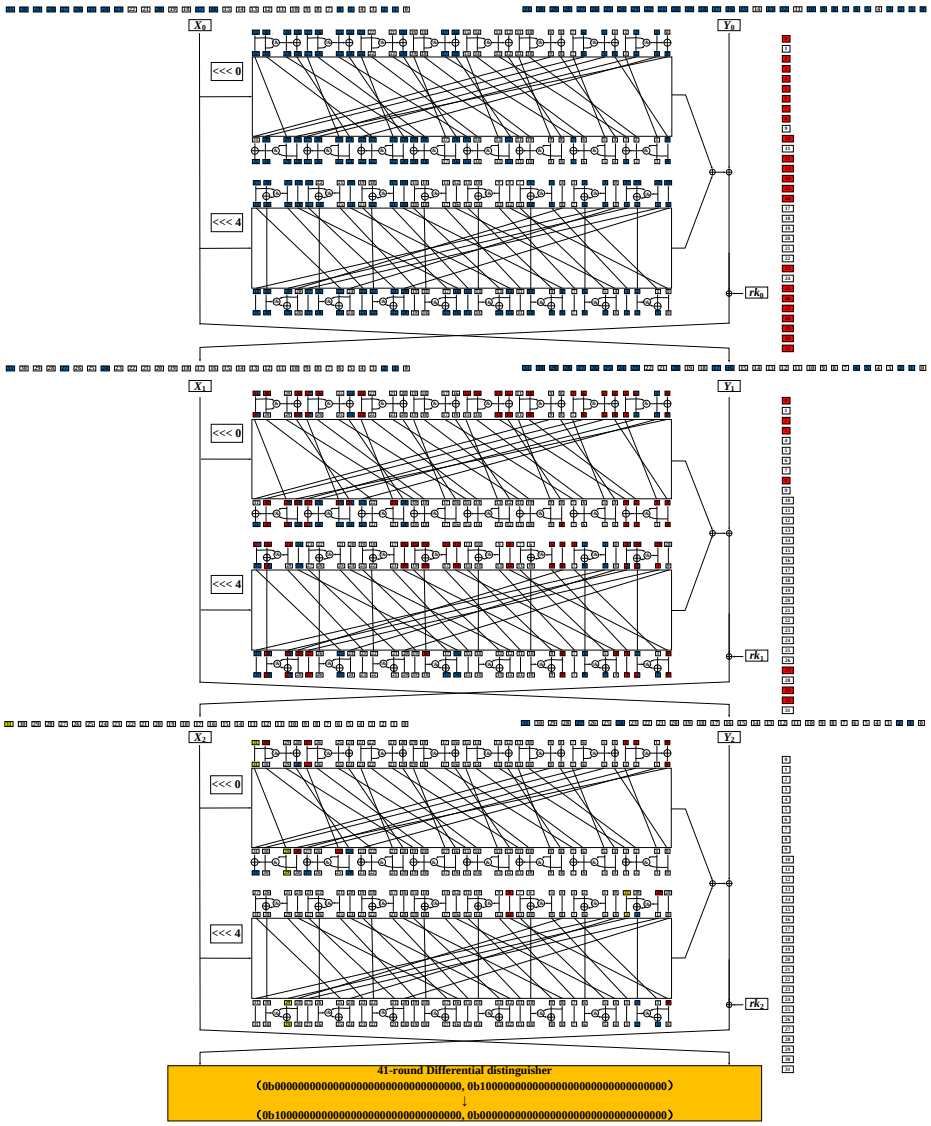

Fig. 10: The head of key recovery attack based on 41-round distinguisher.

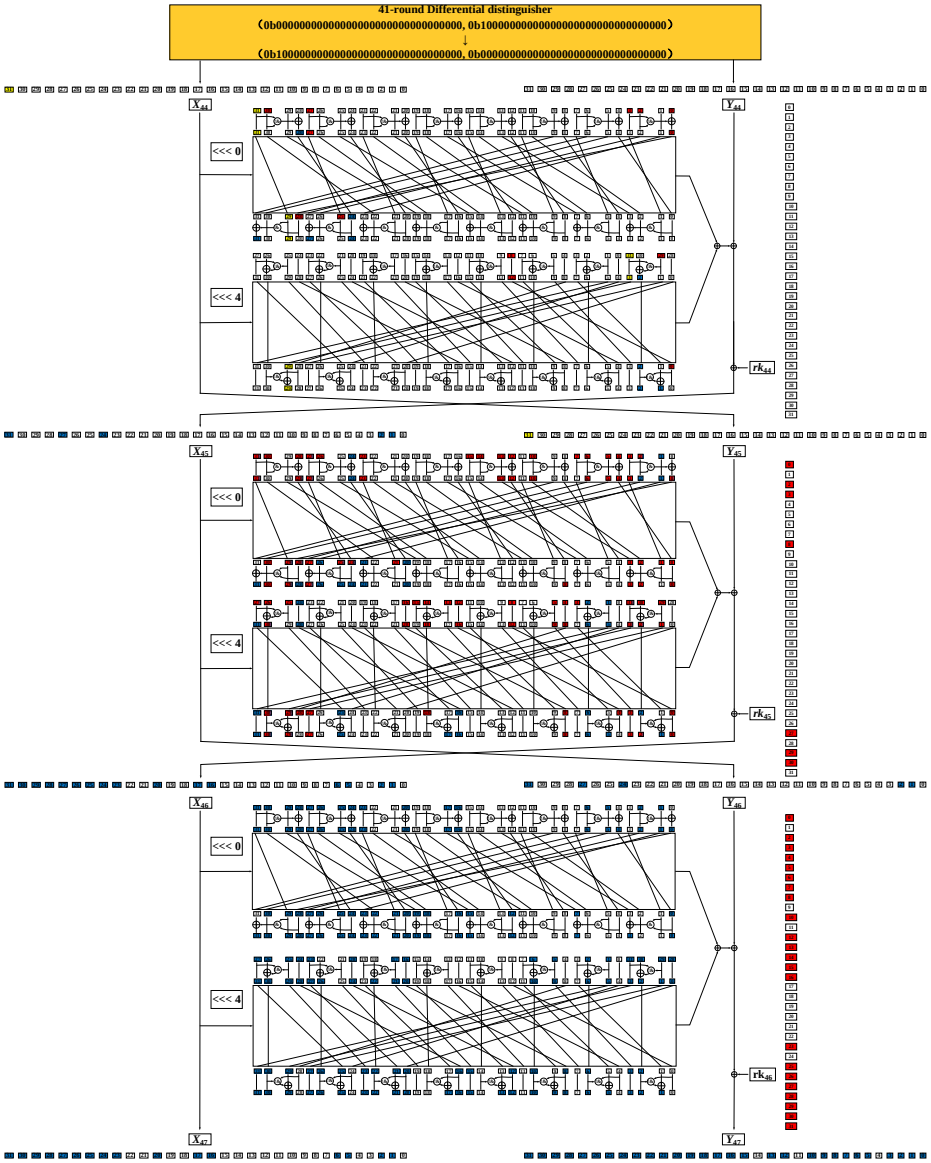

Fig. 11: The tail of key recovery attack based on 41-round distinguisher.

Table 10: Conditions for key recovery and filter probabilities on SAND-2.

Condition	Filter	Probability
(C1)	$\Delta X_{47}[0, 3\text{-}4, 7\text{-}15, 18\text{-}19, 21\text{-}22] \| \Delta Y_{47}[4, 11, 14] = 0$	2^{-19}
(C2)	$\Delta X_1[0, 3, 5\text{-}10, 12\text{-}13, 15\text{-}23, 25\text{-}26, 28\text{-}30] = 0$	2^{-24}
(C3)	$\Delta Y_{46}[0, 3, 5\text{-}10, 12\text{-}13, 15\text{-}23, 25\text{-}26, 28\text{-}30] = 0$	2^{-24}
(C4)	$\Delta X_2[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30] = 0, \Delta X_2[31] = 1$	2^{-16}
(C5)	$\Delta Y_{45}[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30] = 0, \Delta Y_{45}[31] = 1$	2^{-16}
(C6)	$\Delta X_3[1\text{-}2, 24, 27, 31] = 0$	2^{-5}
(C7)	$\Delta Y_{44}[1\text{-}2, 24, 27, 31] = 0$	2^{-5}

3. For each pair $P = (X_0, Y_0)$ and $P' = (X'_0, Y'_0)$, we first calculate $\Delta X_1 = X_1 \oplus X'_1$ without guessing any key bits and check the condition (C2), then $N_1 \cdot 2^{-19} \cdot 2^{-24}$ pairs will be left. We view this operation as one-round encryption, thus the time complexity is about $2 \cdot N_1 \cdot 2^{-19} \times 1/47$ full round encryptions.
4. For each pair $C = (X_{47}, Y_{47})$ and $C' = (X'_{47}, Y'_{47})$, we can calculate ΔY_{46} without guessing key bits and check the condition (C3), then $N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24}$ pairs will be left. Similarly, the time complexity is $2 \cdot N_1 \cdot 2^{-19} \cdot 2^{-24} \times 1/47$ full round encryptions.
5. Guess 22 bits of rk_0. We can compute $X_2[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30, 31]$ and $X'_2[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30, 31]$ for each possible 22-bit subkey value $rk_0[0, 2\text{-}8, 10, 12\text{-}16, 23, 25\text{-}31]$ and check the condition (C4), then $N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{-16}$ pairs will be left. This step involves 18 S-box operation, thus the time complexity is $2 \cdot N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{22} \times 18 \times 1/32 \times 1/47$ full round encryptions.
6. Guess 22 bits of rk_{46}. We can compute $Y_{45}[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30, 31]$ and $Y'_{45}[1\text{-}2, 5\text{-}6, 16\text{-}17, 20, 23\text{-}30, 31]$ for each possible 22-bit subkey value $rk_{46}[0, 2\text{-}8, 10, 12\text{-}16, 23, 25\text{-}31]$ and check the condition (C5), then $N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{-16} \cdot 2^{-16}$ pairs will be left. Similarly, the time complexity is $2 \cdot N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{-16} \cdot 2^{22} \times 18 \times 1/32 \times 1/47$ full round encryptions.
7. Guess 7 bits of rk_1. We can compute $X_3[1\text{-}2, 24, 27, 31]$ and $X'_3[1\text{-}2, 24, 27, 31]$ for each possible 7-bit subkey value $rk_1[0, 2\text{-}3, 8, 27, 29\text{-}30]$ and check the condition (C6), then $N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^{-5}$ pairs will be left. This step involves 6 S-box operation, thus the time complexity is $2 \cdot N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^{7} \times 6 \times 1/32 \times 1/47$ full round encryptions.
8. Guess 7 bits of rk_{45}. We can compute $\Delta Y_{44}[1\text{-}2, 24, 27, 31]$ for each possible 7-bit subkey value $rk_{45}[0, 2\text{-}3, 8, 27, 29\text{-}30]$ and check the condition (C7), then $N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^{-5} \cdot 2^{-5}$ pairs will be left. Similarly, the time complexity is $2 \cdot N_1 \cdot 2^{-19} \cdot 2^{-24} \cdot 2^{-24} \cdot 2^{-16} \cdot 2^{-16} \cdot 2^{-5} \cdot 2^{7} \times 6 \times 1/32 \times 1/47$ full round encryptions.
9. Then do exhaustive search for all keys which correspond to the guessed 58-bit subkey bits against a maximum of two plaintext-ciphertext pairs.

Complexity Analysis. We set a counter to record the number of right pairs that validate the input and output differences of the 41-round distinguisher. With the analysis above, for random key guesses, the number of right pairs is

about $N_1 \cdot 2^{-109}$. For the right key guess, the number of right pairs is expected to be $N_1 \cdot 2^{-45} \cdot 2^{-51.13}$, where 2^{-45} is the probability of the difference of plaintext to the head of the distinguisher and $2^{-51.13}$ is the probability of the distinguisher. In order to get higher success probability we set the number of right pair μ is 2 and the signal-to-noise ratio $S_N = \frac{N_1 \cdot 2^{-45} \cdot 2^{-51.13}}{N_1 \cdot 2^{-109}} = 2^{12.87}$. So the pairs N_1 is $2^{97.13}$ and corresponding N_s is $2^{8.13}$. Thus the data requirement of the attack is $2^{8.13} \cdot 2^{45} = 2^{53.13}$ chosen plaintexts. As we leave only one best candidate, that means advantage a is 58. So the time complexity of this attack can be computed as follows.

$$2^{8.13} \cdot 2^{45} + N_1 \cdot 2^{-23.55} + (1 + 2^{70}) \times (1 + 2^{-64}),$$

where $(1 + 2^{70}) \times (1 + 2^{-64})$ denotes the time complexity of step 9. Then, the time complexity of this attack is about $2^{73.70}$ full round encryptions and the success probability is $P_s = 90.65\%$ for the whole attack. Since we should record the right pairs, the memory complexity of this attack is roughly $2^{53.13}$.