Distinguishing and Key Recovery Attacks on the Reduced-Round SNOW-V

Jin Hoki[1], Takanori Isobe[1,2,3], Ryoma Ito[2], Fukang Liu[1], Kosei Sakamoto[1]

[1] University of Hyogo, Japan.
takanori.isobe@ai.u-hyogo.ac.jp
[2] National Institute of Information and Communications Technology, Japan.
itorym@nict.go.jp
[3] PRESTO, Japan Science and Technology Agency, Japan.

Abstract. This paper proposes distinguishing and key recovery attacks on the reduced-round versions of the SNOW-V stream cipher. First, we construct a MILP model to search for integral characteristics using the division property, and find the best integral distinguisher in the 3-, 4-, and 5-round versions with time complexities of 2^8, 2^{16}, and 2^{48}, respectively. Next, we construct a bit-level MILP model to efficiently search for differential characteristics, and find the best differential characteristics in the 3- and 4-round versions. These characteristics lead to the 3- and 4-round differential distinguishers with time complexities of 2^{48} and 2^{103}, respectively. Then, we consider single-bit and dual-bit differential cryptanalysis, which is inspired by the existing study on Salsa and ChaCha. By carefully choosing the IV values and differences, we observe the best bit-wise differential biases with $2^{-1.733}$ and $2^{-17.934}$ in the 4- and 5-round versions, respectively. This is feasible to construct a very practical distinguisher with a time complexity of $2^{4.466}$ for the 4-round version, and a distinguisher with a time complexity of at least $2^{36.868}$ for the 5-round version. Finally, we improve the existing differential attack based on probabilistic neutral bits, which is also inspired by the existing study on Salsa and ChaCha. As a result, we present the best key recovery attack on the 4-round version with a time complexity of $2^{153.97}$ and data complexity of $2^{26.96}$. Consequently, we significantly improve the existing best attacks in the initialization phase by the designers.

Keywords: SNOW · Stream cipher · 5G · Integral attack · Differential attack · Probabilistic Neutral Bits (PNB)

1 Introduction

1.1 Background

SNOW-V, which is a new variant of a family of SNOW stream ciphers, was proposed for a standard encryption scheme for the 5G mobile communication system in 2019 by Ekdahl et al. [5]. To achieve the strong security requirements by the 3GPP standardization organization for the 5G system, SNOW-V provides

a 256-bit security level against key recovery attacks with a 256-bit key and 128-bit IV, while the claimed security of distinguishing attacks is only 2^{64}, i.e., the length of keystreams is limited to at most 2^{64} and also for a fixed key, the number of different keystreams should be less than 2^{64}.

SNOW-V consists of a Linear Feedback Shift Register (LFSR) and Finite State Machine (FSM). The overall structure of SNOW-V follows the design strategy of SNOW 2.0 and SNOW-3G. It takes advantage of AES-NI and some SIMD operations for efficient implementation in high-end software environments. Each round has two AES-round operations to update the states of the FSM. As a result, SNOW-V achieves very impressive performance in software, e.g., 58 Gbps for a long message, which is almost six times faster than that of SNOW-3G.

Regarding the security analysis of SNOW-V, the designers evaluated the security of division-property-based cube, time-memory tradeoff, linear/correlation distinguishing, algebraic, and guess-and-determine attacks [5]. Among them, they found a key recovery attack on the 3-round SNOW-V by cube attacks, and concluded that more than four rounds provides sufficient security against these attacks as the division-property-based distinguisher reaches only four rounds of AES [16]. As a third-party evaluation, Jiao et al. proposed a byte-based guess-and-determine attack with a time complexity of 2^{406} [9]. They improved the authors' evaluation, but its cost is still much larger than the exhaustive 256-bit key search. Thus, to the best of our knowledge, the best attack on SNOW-V is the 3-round cube attack by the designers.

1.2 Our Contribution

In this study, we investigate the security of SNOW-V with three attack vectors, namely, integral, differential, and bit-wise differential attacks. These attacks are well-known attacks for stream ciphers. Nevertheless, the designers did not perform the security evaluations for these important attacks. To fill this gap, we evaluate thorough security against these attacks with state-of-the-art search tools and techniques, and we show that these attacks sufficiently improve the previous best attacks with respect to the attacked number of rounds and attack complexity, as shown in Table 1. The details of our attacks are given as follows.

Integral Attack. By using a MILP-aided search method for the division property, we show practical integral distinguishers in the 3/4-round distinguishers with time complexities of 2^8 and 2^{16}. Furthermore, we find a 5-round integral distinguisher with a time complexity of 2^{48} for the initialization of SNOW-V.

Differential Attack. We perform a MILP-aided search for the differential characteristics in the chosen-IV setting where differences are inserted in the IV domain. Specifically, we build a bit-level model for each operation, such as the modular addition, S-box, and linear operations. As a result, we find the 3/4-round differential characteristics with probabilities of 2^{-48} and 2^{-103}, respectively. Although the 4-round distinguishing attack exceeds the data limitation of 2^{64}, it is important to improve the understanding of the security of SNOW-V.

Table 1. Summary of our results.

Attack type	Rounds	Data	Time	Reference
Integral/Distinguisher	3	$2^{8.00}$	$2^{8.00}$	Section 3
Integral/Distinguisher	4	$2^{16.00}$	$2^{16.00}$	Section 3
Integral/Distinguisher	5	$2^{48.00}$	$2^{48.00}$	Section 3
Differential/Distinguisher	3	$2^{48.00}$	$2^{48.00}$	Section 4
Differential/Distinguisher	4	$2^{103.00}$	$2^{103.00}$	Section 4
Differential Bias/Distinguisher	4	$2^{4.47}$	$2^{4.50}$	Section 5
Differential Bias/Distinguisher	5	$2^{36.87}$	$2^{36.87}$	Section 5
Cube/Key Recovery	3	$2^{15.00}$	$2^{255.00}$	[5]
Differential Bias/Key Recovery	4	$2^{26.96}$	$2^{153.97}$	Section 6

Bit-wise Differential Attack. We conduct a single-bit and dual-bit differential attack based on the existing study on the reduced-round Salsa and ChaCha as reported by Choudhuri and Maitra [3]. In addition, we analyze the source code of the LFSR update algorithm in SNOW-V, and suggest that choosing IVs by limiting the domain should suppress the propagation of differences throughout the internal state of SNOW-V. As a result, we find a practical bit-wise differential distinguisher for the 4-round SNOW-V. Surprisingly, it is feasible by only $2^{4.466}$ samples. We further observe the 5-round differential biases, and we present a theoretical distinguisher with time complexity of at least $2^{36.868}$ for the 5-round SNOW-V. No study has been reported on applying the bit-wise differential attack to LFSR-based stream ciphers; thus, in this study, we have demonstrated the effectiveness of the bit-wise differential attack on LFSR-based stream ciphers.

Key Recovery Attack. We apply the differential attack based on probabilistic neutral bits (PNB), which was proposed by Aumasson et al. [2], to a key recovery attack on SNOW-V. To apply an existing attack, it is necessary to perform the backwards computation in the target cipher, but it is difficult to perform this in SNOW-V. To solve this problem, we replace all the backwards computations in the existing attack procedure with forwards computations. As a result, we present a key recovery attack on the 4-round SNOW-V with a time complexity of $2^{153.97}$ and data complexity of $2^{26.96}$. To the best of our knowledge, our attack is the best key recovery attack on the reduced-round SNOW-V.

1.3 Organization of the Paper

The rest of the paper is organized as follows. In Section 2, we briefly describe the specification of the SNOW-V stream cipher. In Section 3, we show the MILP model for searching integral characteristics and provide integral distinguishers for 3, 4, and 5 rounds of SNOW-V. In Section 4, we show the MILP model for

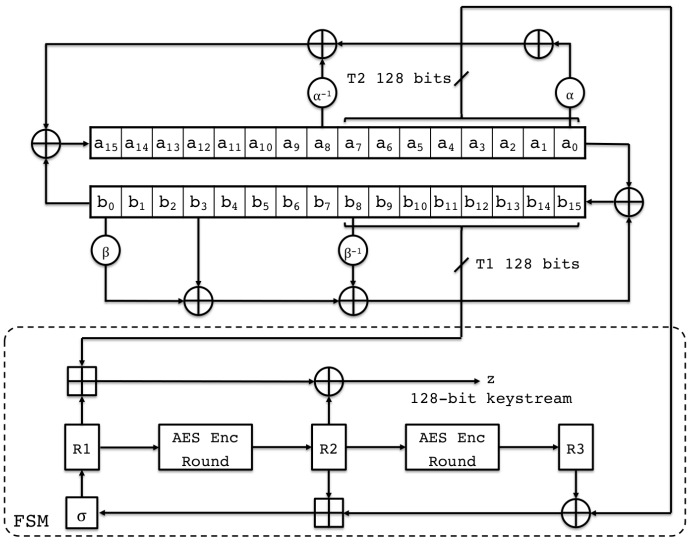

Fig. 1. Overall structure of SNOW-V.

searching differential characteristics and provide differential distinguishers for 3 and 4 rounds of SNOW-V. In Section 5, we introduce the existing cryptanalysis method for bit-wise differential cryptanalysis and present the efficient chosen-IV technique. We then provide bit-wise differential distinguishers for 4 and 5 rounds of SNOW-V. In Section 6, we describe our improvements to the existing differential attack and present the best key recovery attack on the 4-round SNOW-V. Finally, Section 7 concludes the paper.

2 Description of SNOW-V

2.1 Structure

The overall structure of SNOW-V is shown in Figure 1. It consists of a Linear Feedback Shift Register (LFSR) part and Finite State Machine (FSM) part.

The LFSR part takes a circular construction consisting of two shift registers called LFSR-A and LFSR-B, both involving 16 cells with each cell size of 16 bits denoted by $a_{15}, \ldots, a_0$ and $b_{15}, \ldots, b_0$, respectively. Each cell represents an element in $\mathbb{F}_2^{16}$, and the elements of LFSR-A and LFSR-B are generated by the following polynomials in $\mathbb{F}_2[x]$:

$$g^A(x) = x^{16} + x^{15} + x^{12} + x^{11} + x^8 + x^3 + x^2 + x + 1, \quad (1)$$
$$g^B(x) = x^{16} + x^{15} + x^{14} + x^{11} + x^8 + x^6 + x^5 + x + 1. \quad (2)$$

Let $\alpha \in \mathbb{F}_{2^{16}}^A$ be a root of $g^A(x)$ and $\beta \in \mathbb{F}_{2^{16}}^B$ be a root of $g^B(x)$. At time $t \geq 0$, the LFSRs update sequences $(a_{15}^{(t)}, \ldots, a_0^{(t)})$ and $(b_{15}^{(t)}, \ldots, b_0^{(t)})$ using the

following expressions:

$$a_{15}^{(t+1)} = b_0^{(t)} + \alpha a_0^{(t)} + a_1^{(t)} + \alpha^{-1} a_8^{(t)} \mod g^A(\alpha), \tag{3}$$

$$a_i^{(t+1)} = a_{i+1}^{(t)}, \tag{4}$$

$$b_{15}^{(t+1)} = a_0^{(t)} + \beta b_0^{(t)} + a_3^{(t)} + \beta^{-1} b_8^{(t)} \mod g^B(\beta), \tag{5}$$

$$b_i^{(t+1)} = b_{i+1}^{(t)}, \tag{6}$$

for $i = 0, \ldots, 14$. The LFSRs update the internal state eight times in a single step, i.e., 16 cells of the total 32 cells in the LFSR part can be updated in a single step, and the two taps $T1$ and $T2$ will have the following new values:

$$T1^{(t)} = (b_{15}^{(8t)}, \ldots, b_8^{(8t)}), \tag{7}$$

$$T2^{(t)} = (a_7^{(8t)}, \ldots, a_0^{(8t)}). \tag{8}$$

The FSM part takes the two taps, $T1$ and $T2$, from the LFSR part as the inputs and generates a 128-bit keystream block $z^{(t)}$ at time $t \geq 0$ as the output. It consists of three 128-bit registers $R1$, $R2$, and $R3$. The symbol $\oplus$ denotes a bit-wise XOR operation, and the symbol $\boxplus_{32}$ denotes parallel application of four additions modulo 2^{32}. The four 32-bit parts of the 128-bit words are added with carry, but the carry does not propagate from a lower 32-bit word to a higher one. At time $t \geq 0$, the FSM first outputs the keystream block, $z^{(t)}$, using the following expression:

$$z^{(t)} = (R1^{(t)} \boxplus_{32} T1^{(t)}) \oplus R2^{(t)}. \tag{9}$$

Then, registers $R2$ and $R3$ are updated throughout a full AES encryption round function as `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey`, which are denoted by $\texttt{AES}^R(IN, KEY)$ with a 128-bit input block IN and a roundkey KEY. The three registers are updated by the following expressions:

$$R1^{(t+1)} = \sigma(R2^{(t)} \boxplus_{32} (R3^{(t)} \oplus T2^{(t)})), \tag{10}$$

$$R2^{(t)} = \texttt{AES}^R(R1^{(t)}, 0), \tag{11}$$

$$R3^{(t)} = \texttt{AES}^R(R2^{(t)}, 0), \tag{12}$$

where σ is a byte-oriented permutation given by

$$\sigma = [0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15]. \tag{13}$$

2.2 Initialization

Let $K = (k_{15}, \ldots, k_0)$ denote a 256-bit key and $IV = (iv_7, \ldots, iv_0)$ denote a 128-bit initialization vector (IV), where each k_i and iv_j are 16-bit vectors for $0 \leq i \leq 15$ and $0 \leq j \leq 7$, respectively. The initialization begins with loading the key and IV into the LFSRs and setting zero into the three registers using the following expressions:

$$(a_{15}, \ldots, a_0) = (k_7, \ldots, k_0, iv_7, \ldots, iv_0), \tag{14}$$

$$(b_{15}, \ldots, b_0) = (k_{15}, \ldots, k_8, 0, \ldots, 0), \tag{15}$$
$$R1 = 0,\ R2 = 0,\ R3 = 0. \tag{16}$$

The initialization consists of r steps ($r = 16$ in the original version), where the structure is updated in the same way as in the keystream generation, with the exception that the 128-bit keystream block z is not an output but is XORed into the LFSR-A to positions $(a_{15}, \ldots, a_8)$ in every step. Additionally, at the two last steps of the initialization, the 256-bit key is loaded into the register $R1$ using the following expressions:

$$R1^{(r-2)} = R1^{(r-2)} \oplus (k_7, \ldots, k_0), \tag{17}$$
$$R1^{(r-1)} = R1^{(r-1)} \oplus (k_{15}, \ldots, k_8), \tag{18}$$

where time $t = r - 1$ denotes the last step of the initialization.

The designers limited the length of the keystream to a maximum of 2^{64} bits for a single key-IV pair and the number of different IVs to a maximum of 2^{64} for each key.

3 MILP-aided Integral Distinguisher

In this section, we explore the security of SNOW-V against integral attacks. To efficiently search for integral distinguishers in the initialization phase of SNOW-V, we exploit the division property proposed by Todo [15]. Specifically, we utilize the MILP-based method [17] to evaluate the propagation of the bit-based division property [16].

3.1 The MILP Model

In this part, we describe how to construct the linear inequalities to model the propagation of the division property for SNOW-V. First, we will show the constraints for the propagation of the bit-based division property through `COPY`, `XOR`, and `AND` operations based on the work by Xiang et al. [17]. Then, we elaborate the MILP model for SNOW-V based on these constraints.

To find an integral distinguisher with the division property with MILP, we do not need to optimize the objective function. Instead, we only need to confirm whether the constructed MILP model is feasible or not, because we search the properties such that the output is balanced or not by bit-wise. If it is infeasible, an integral distinguisher can be obtained.

Xiang et al. first proposed the modeling method [17] for the propagation of the bit-based division property through `COPY`, `XOR`, and `AND` operations. Then, Sun et al. generalized these models [13] as specified below, which will be the components in our MILP model for SNOW-V.

$$\text{MILP Model of } \texttt{COPY}\ [13]: \begin{cases} \mathcal{M}.var \leftarrow a, b_1, \ldots, b_m \text{ as binary.} \\ \mathcal{M}.con \leftarrow a + b_1 + \cdots + b_m = 0. \end{cases}$$

$$\text{MILP Model of } \texttt{XOR}\ [13]: \begin{cases} \mathcal{M}.var \leftarrow a_1, \ldots, a_m, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow a_1 + \cdots + a_m + b = 0. \end{cases}$$

$$\text{MILP Model of } \texttt{AND}\ [17]: \begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow b - a_1 \geq 0, \\ \mathcal{M}.con \leftarrow b - a_2 \geq 0, \\ \mathcal{M}.con \leftarrow b - a_1 - a_2 \leq 0. \end{cases}$$

The pseudo code of our MILP model for SNOW-V is displayed in Algorithm 1, where R denotes the number of rounds in the initialization phase and the explanations for `load`, `funcADD`, `funcAES`, `sigma`, and `funcLFSR` are given below.

`load`. K and IV are loaded into internal states.

`funcADD`. This function is a model for the 32-bit modular addition. We use the modeling method proposed by Sun et al. [14] with `COPY`, `XOR`, and `AND`.

`funcAES`. This function consists of SubBytes, ShiftRow, MixColumns, and AddRoundKey of AES. For the modeling of the S-box, we use the modeling method proposed in [17]. Logic Friday [4] is utilized to generate the constraints for the S-box. Thus, we obtain 241 linear inequalities to model the S-box of AES. For the modeling of MixColumns, we use the modeling method proposed in [13]. Specifically, the 4×4 MDS matrix over the filed $\mathbb{F}_2^8$ is converted to a 32×32 binary matrix over the field, $\mathbb{F}_2$ [12]. Then, we construct the model for MixColumn with `COPY` and `XOR`. Thus, 64 linear inequalities can be used to model the MDS matrix used in AES.

`sigma`. This function is used to permute the state in a byte-wise way as described in Section 2.

`funcLFSR`. There are the operations of α, α^{-1}, β, β^{-1}, and `XOR`. It is a linear transformation; thus, the division property of the input and the output are constant. Hence, we can use the method from Sun et al. [13], and α, α^{-1}, β, and β^{-1} are each represented with a 16×16 matrix over field $\mathbb{F}_2$, and we obtain 64 linear inequalities.

3.2 Our Search and Results

Since there are a total of 2^{128} patterns for IV, it is computationally infeasible to take all of them into account when searching for integral distinguishers. Thus, we use a 3-step approach to efficiently find the integral distinguisher. As an explanation of our method, `a`, `c`, `b`, and `u` represent an active bit, a constant bit, a balanced bit, and an unknown bit, respectively. In addition, $\mathcal{A}$, $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{U}$ denote an active byte, a constant byte, a balanced byte, and an unknown byte, respectively. Our search used Gurobi optimization 9.0 [8] as the solver with a 48-core Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz for our experiments.

Step 1. We try to find the longest integral distinguisher by setting the 128-bit IV as all $\mathcal{A}$.

Algorithm 1 MILP model of division property for SNOW-V

procedure SNOWVcore(round R)
  for $out = 0$ to 127 **do**
    Prepare an empty MILP model $\mathcal{M}$
    $\mathcal{M}.var \leftarrow K_j, IV_j$ for $j \in \{0, \ldots, 127\}$
    $\mathcal{M}.var \leftarrow S_j^r$ for $j \in \{0, \ldots, 511\}$ and for $r \in \{0, \ldots, R+1\}$
    $\mathcal{M}.var \leftarrow R1_j^r, R2_j^r, R3_j^r$ for $j \in \{0, \ldots, 127\}$ and for $r \in \{0, \ldots, R+1\}$
    $\mathcal{M}.var \leftarrow Z_j^r$ for $j \in \{0, \ldots, 127\}$ and for $r \in \{0, \ldots, R\}$
    $\mathcal{M}.con \leftarrow \boldsymbol{K} = 0$
    $(\mathcal{M}, \boldsymbol{IV}) \Leftarrow$ *initial division property* (Section 3.2)
    $(\mathcal{M}, \boldsymbol{S}^0) = \texttt{load}(\mathcal{M}, \boldsymbol{K}, \boldsymbol{IV})$
    $\mathcal{M}.con \leftarrow \boldsymbol{R1}^0 = 0, \boldsymbol{R2}^0 = 0, \boldsymbol{R3}^0 = 0$
    for $r = 0$ to R **do**
      $(\mathcal{M}, \boldsymbol{T2}^r, S^{r,0}_{128,\ldots,255}) = \texttt{COPY}(\mathcal{M}, S^r_{128,\ldots,255})$
      $(\mathcal{M}, \boldsymbol{T1}^r, S^{r,0}_{256,\ldots,383}) = \texttt{COPY}(\mathcal{M}, S^r_{256,\ldots,383})$
      $(\mathcal{M}, \boldsymbol{X}^r_{R1}, \boldsymbol{Y}^r_{R1}) = \texttt{COPY}(\mathcal{M}, \boldsymbol{R1}^r)$
      $(\mathcal{M}, \boldsymbol{X}^r_{R2}, \boldsymbol{Y}^r_{R2}, \boldsymbol{W}^r_{R2}) = \texttt{COPY}(\mathcal{M}, \boldsymbol{R2}^r)$
      $\mathcal{M}.con \leftarrow \boldsymbol{S}^{r,0}_{0,\ldots,127} = \boldsymbol{S}^r_{0,\ldots,127}$
      $\mathcal{M}.con \leftarrow \boldsymbol{S}^{r,0}_{384,\ldots,511} = \boldsymbol{S}^r_{384,\ldots,511}$
      $(\mathcal{M}, \boldsymbol{U}^r) = \texttt{funcADD}(\mathcal{M}, \boldsymbol{T1}^r, \boldsymbol{X}^r_{R1})$
      $(\mathcal{M}, \boldsymbol{Z}^r) = \texttt{XOR}(\mathcal{M}, \boldsymbol{U}^r, \boldsymbol{X}^r_{R2})$
      $(\mathcal{M}, \boldsymbol{R2}^{r+1}) = \texttt{funcAES}(\mathcal{M}, \boldsymbol{Y}^r_{R1})$
      $(\mathcal{M}, \boldsymbol{R3}^{r+1}) = \texttt{funcAES}(\mathcal{M}, \boldsymbol{Y}^r_{R2})$
      $(\mathcal{M}, \boldsymbol{V}^r) = \texttt{XOR}(\mathcal{M}, \boldsymbol{T2}^r, \boldsymbol{R3}^r)$
      $(\mathcal{M}, \boldsymbol{tmp}^r) = \texttt{funcADD}(\mathcal{M}, \boldsymbol{V}^r, \boldsymbol{W}^r_{R2})$
      $(\mathcal{M}, \boldsymbol{R1}^{r+1}) = \texttt{sigma}(\mathcal{M}, \boldsymbol{tmp}^r)$
      for $i = 0$ to 7 **do**
        $(\mathcal{M}, \boldsymbol{S}^{r,i+1}) = \texttt{funcLFSR}(\mathcal{M}, \boldsymbol{S}^{r,i})$
      if $r = R$ **then**
        $\mathcal{M}.con \leftarrow \boldsymbol{S}^{r+1}_{0,\ldots,511} = \boldsymbol{S}^{r,8}_{0,\ldots,511}$
      else
        $(\mathcal{M}, \boldsymbol{S}^{r+1}_{0,\ldots,127}) = \texttt{XOR}(\mathcal{M}, \boldsymbol{S}^{r,8}_{0,\ldots,127}, \boldsymbol{Z}^r)$
        $\mathcal{M}.con \leftarrow \boldsymbol{S}^{r+1}_{128,\ldots,511} = \boldsymbol{S}^{r,8}_{128,\ldots,511}$
    $\mathcal{M}.con \leftarrow \boldsymbol{S}^{R+1} = 0$
    $\mathcal{M}.con \leftarrow \boldsymbol{R1}^{R+1} = 0, \boldsymbol{R2}^{R+1} = 0, \boldsymbol{R3}^{R+1} = 0$
    for $j = 0$ to 127 **do**
      if $j = out$ **then**
        $\mathcal{M}.con \leftarrow Z_j = 1$
      else
        $\mathcal{M}.con \leftarrow Z_j = 0$

Step 2. To reduce the data complexity, we consider the case where there is at least one byte in IV assigned to $\mathcal{C}$ and at least one byte assigned to $\mathcal{A}$. When 16-byte input is all $\mathcal{A}$, it is the same as Step 1. Also, when 16-byte input is all $\mathcal{C}$, the outputs becomes constants. Thus, these two patterns can be omitted. As a result, there are $2^{16} - 2$ such patterns in total.

Table 2. 3-Round Integral Distinguisher of SNOW-V

iv_7	cccccccc cccccccc
iv_6	cccccccc cccccccc
iv_5	cccccccc cccccccc
iv_4	cccccccc cccccccc
iv_3	aaaaaaaa cccccccc
iv_2	cccccccc cccccccc
iv_1	cccccccc cccccccc
iv_0	cccccccc cccccccc
z	uuuuuuuu uuuuuuuu uuuuuuuu bbbbbbbb
	uuuuuuuu uuuuuuuu uuuuuuuu bbbbbbbb
	uuuuuuuu uuuuuuuu bbbbbbbb bbbbbbbb
	uuuuuuuu uuuuuuuu bbbbbbbb bbbbbbbb

Step 3. We utilize the method [7] to reduce the data complexity. In [7], a is only assigned to the MSB of each byte. First, we consider the case when there is only one active bit and the total number of such patterns is $\binom{16}{1}$. Then, we increase the number of a if we can find an integral distinguisher, i.e., consider the case when there are $2, 3, 4, \ldots, 16$ active bits because IV is a 16-byte value. Thus, a total of $2^{16} - 1$ patterns is taken into account in our search.

Our search found integral distinguishers in 3- and 4-round distinguishers with time complexities of 2^8 and 2^{16}, as shown in Tables 2 and 3. Moreover, we can find a 5-round integral distinguisher for the initialization phase of SNOW-V, as shown in Table 4. Specifically, when iv_7, iv_6, iv_4 and iv_0 is constant, the least significant byte of iv_2 and iv_1 is constant, and the remaining bytes of IV take all the possible 2^{48} values, we can compute the sum of the keystreams, z, generated by these 2^{48} different IV; thus, the sum in each of the least two significant bits of z is always zero.

4 MILP-aided Differential Distinguisher

In this section, we describe our investigation of the resistance of SNOW-V against differential attacks.Specifically, we focus on the initialization phase and our aim is to find differential characteristics with a probability higher than 2^{-128} using a MILP-based method [1, 6] as the IV size where differences of 128 bits can be inserted.

According to the specification of SNOW-V, it can be observed that there are 32 AES S-boxes and 8 modular additions (modulo 2^{32}) used for the 1-round state update, which are the only components where the difference transitions are probabilistic.

Table 3. 4-Round Integral Distinguisher of SNOW-V

iv_7	aaaaaaaa cccccccc
iv_6	cccccccc cccccccc
iv_5	aaaaaaaa cccccccc
iv_4	cccccccc cccccccc
iv_3	cccccccc cccccccc
iv_2	cccccccc cccccccc
iv_1	cccccccc cccccccc
iv_0	cccccccc cccccccc
z	uuuuuuuu uuuuuuuu uuuuuuuu uuuuuuuu
	uuuuuuuu uuuuuuuu uuuuuuuu uuuuuuuu
	uuuuuuuu uuuuuuuu uuuuuuuu bbbbbbbb
	uuuuuuuu uuuuuuuu uuuuuubb bbbbbbbb

Table 4. 5-Round Integral Distinguisher of SNOW-V

iv_7	cccccccc cccccccc
iv_6	cccccccc cccccccc
iv_5	aaaaaaaa aaaaaaaa
iv_4	cccccccc cccccccc
iv_3	aaaaaaaa aaaaaaaa
iv_2	aaaaaaaa cccccccc
iv_1	aaaaaaaa cccccccc
iv_0	cccccccc cccccccc
z	uuuuuuuu uuuuuuuu uuuuuuuu uuuuuuuu
	uuuuuuuu uuuuuuuu uuuuuuuu uuuuuuuu
	uuuuuuuu uuuuuuuu uuuuuuuu uuuuuuuu
	uuuuuuuu uuuuuuuu uuuuuuuu uuuuuubb

4.1 The MILP Model

Here, we explain the details of our MILP modeling for searching differential characteristics of SNOW-V. Throughout this paper, $\mathcal{M}.var$, $\mathcal{M}.con$, and $\mathcal{M}.obj$ represent the variables, the constraints and the objective function in the MILP model, respectively.

Because the operations used in SNOW-V include `XOR`, SubBytes, ShiftRows, MixColumns, modular addition, α, α^{-1}, β, β^{-1}, and `sigma` (the byte-wise permutation), to construct an accurate model to describe the bit-wise difference in propagation using these components, it is necessary to construct the corresponding linear inequalities for each of them.

Algorithm 2 MILP model of differential characteristics for SNOW-V

procedure `SNOWVcore`(round R)
  Prepare an empty MILP model $\mathcal{M}$
  $\mathcal{M}.var \leftarrow K_j, IV_j$ for $j \in \{0, \ldots, 127\}$
  $\mathcal{M}.var \leftarrow S_j^r$ for $j \in \{0, \ldots, 511\}$ and for $r \in \{0, \ldots, R\}$
  $\mathcal{M}.var \leftarrow R1_j^r, R2_j^r, R3_j^r$ for $j \in \{0, \ldots, 127\}$ and for $r \in \{0, \ldots, R\}$
  $\mathcal{M}.var \leftarrow Z_j^r$ for $j \in \{0, \ldots, 127\}$ and for $r \in \{0, \ldots, R\}$
  $\mathcal{M}.con \leftarrow \boldsymbol{K} = 0$
  $\mathcal{M}.con \leftarrow \boldsymbol{IV} \geq 1$
  $(\mathcal{M}, \boldsymbol{S}^0) = \texttt{load}(\mathcal{M}, \boldsymbol{K}, \boldsymbol{IV})$
  $\mathcal{M}.con \leftarrow \boldsymbol{R1}^0 = 0, \boldsymbol{R2}^0 = 0, \boldsymbol{R3}^0 = 0$
  for $r = 0$ to $R - 1$ **do**
    $(\mathcal{M}, \boldsymbol{U}^r) = \texttt{funcADD}(\mathcal{M}, S^r_{256,\ldots,383}, \boldsymbol{R1}^r)$
    $(\mathcal{M}, \boldsymbol{Z}^r) = \texttt{XOR}(\mathcal{M}, \boldsymbol{U}^r, \boldsymbol{R2}^r)$
    $(\mathcal{M}, \boldsymbol{R2}^{r+1}) = \texttt{funcAES}(\mathcal{M}, \boldsymbol{R1}^r)$
    $(\mathcal{M}, \boldsymbol{R3}^{r+1}) = \texttt{funcAES}(\mathcal{M}, \boldsymbol{R2}^r)$
    $(\mathcal{M}, \boldsymbol{V}) = \texttt{XOR}(\mathcal{M}, \boldsymbol{T2}, \boldsymbol{R3})$
    $(\mathcal{M}, \boldsymbol{tmp}) = \texttt{funcADD}(\mathcal{M}, \boldsymbol{V}, \boldsymbol{R2}^r)$
    $(\mathcal{M}, \boldsymbol{R1}^{r+1}) = \texttt{sigma}(\mathcal{M}, \boldsymbol{tmp})$
    for $i = 0$ to 7 **do**
      $(\mathcal{M}, \boldsymbol{S}^{r,i+1}) = \texttt{funcLFSR}(\mathcal{M}, \boldsymbol{S}^{r,i})$
    $(\mathcal{M}, \boldsymbol{S}^{r+1}_{0,\ldots,127}) = \texttt{XOR}(\mathcal{M}, \boldsymbol{S}^{r,8}_{0,\ldots,127}, \boldsymbol{Z}^r)$
    $\mathcal{M}.con \leftarrow \boldsymbol{S}^{r+1}_{128,\ldots,511} = \boldsymbol{S}^{r,8}_{128,\ldots,511}$
  $(\mathcal{M}, \boldsymbol{U}^r) = \texttt{funcADD}(\mathcal{M}, S^R_{256,\ldots,383}, \boldsymbol{R1}^R)$
  $(\mathcal{M}, \boldsymbol{Z}^R) = \texttt{XOR}(\mathcal{M}, \boldsymbol{U}^r, \boldsymbol{R2}^R)$
  $\mathcal{M}.obj \leftarrow Minimize(DCP)$

`XOR`. The following linear inequalities can be used to model $x_2 = x_0 \oplus x_1$

$$\text{MILP Model of } \texttt{XOR}: \begin{cases} x_0, x_1, x_2 \text{ as binary.} \\ -x_0 - x_1 - x_2 \geq -2, \\ -x_0 + x_1 + x_2 \geq 0, \\ x_0 - x_1 + x_2 \geq 0, \\ x_0 + x_1 - x_2 \geq 0. \end{cases}$$

`load`. K and IV are loaded into internal states.

`funcAES`. This function consists of the SubBytes, ShiftRows, MixColumns and AddRoundKey of AES. As proposed in ref. [1], we utilize Logic Friday [4] to automatically generate the linear inequalities for the AES S-box. There are a total of 8302 linear inequalities needed to describe the difference distribution table of the AES S-box. Because it is a linear transform, we could write the 4×4 MDS matrix as a 32×32 binary matrix. Using this method, modeling MixColumns is equivalent to modeling several $\oplus$ operations.

`funcADD`. As proposed in ref. [6], we obtain 407 linear inequalities to model the 32-bit modular addition.

`sigma`. This function is used to permute the state in a byte-wise way, as described in Section 2.

`funcLFSR`. This function consists of α, α^{-1}, β, β^{-1}, and `XOR`. Because α, α^{-1}, β, and β^{-1} are all linear transformations as well, we can derive the equivalent 16×16 binary matrix for all of them, which can be simply modeled by considering the linear inequalities for the $\oplus$ operation.

To search for the best differential characteristic, we minimize the objective function, as follows:

$$\sum_{r=0}^{R-1} \left(7 \sum_{m=0}^{31} A_m^r + \sum_{m=0}^{7} \sum_{n=1}^{31} M_m^r[n] \right) + \sum_{m=0}^{4} \sum_{n=1}^{31} M_m^R[n].$$

A_i^r denotes the variable of S-box input, and $M_m^r[n] = \neg eq(\alpha_m^r[n], \beta_m^r[n], \gamma_m^r[n])$, i.e., $eq(\alpha_m^r[n], \beta_m^r[n], \gamma_m^r[n]) = 1$ is same as $\alpha_m^r[n] = \beta_m^r[n] = \gamma_m^r[n]$. $\alpha_m^r[n]$ and $\beta_m^r[n]$ denote the variables of modular addition inputs, and $\gamma_m^r[n]$ denotes the variables of modular addition output, and, $\alpha_m^r[0], \beta_m^r[0], \gamma_m^r[0]$ are the most significant bit, respectively. We consider the differential probability of the modular addition according to [10, 6], and we consider that the differential probability of AES S-box with 2^{-7} because we consider the worst case in the S-box. Algorithm 2 shows the MILP model of differential characteristics for SNOW-V.

4.2 Our Search and Results

In our search, a difference will only be inserted in IV, i.e., we do not consider related-key differential characteristics. First, we try to consider no constraints on IV differences, and could obtain the optimal differential characteristics for up to 3-initialization rounds in feasible time, where we used Gurobi optimization 9.0 [8] as the solver with a 48-core Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz for our experiments.

The search results are displayed in Table 5. For 3-rounds, the best differential probability of a single trail is estimated as 2^{-48} as shown in Table 6. It implies that a distinguishing attacks on 3-rounds is feasible with 2^{48} chosen IVs. Since we search the whole space of IV, these differential probabilities are optimal for 1- to 3-rounds.

To search for more rounds, we constrain that the hamming weight of the IV difference is one because the above optimal characteristic of 1- to 3-rounds are started from the IV difference whose hamming weight one. In this way, we search for differential characteristics up to 4-initialization rounds. As a result, we found a differential characteristic with probability of 2^{-103} as shown in Table 7. To mount the attack using this characteristic, it requires 2^{103} chosen IVs. So, it exceeds the data limitations for a fixed key of 2^{64}. However, we believe that it is meaningful for deeply understanding the security of SNOW-V, e.g., it might be feasible in the weak-key setting.

Table 5. Differential characteristic probability (DCP) for the reduced initialization round

Rounds	1	2	3	4
Optimal DCP (whole IV space)	1	2^{-8}	2^{-48}	–
Optimal DCP (hamming weight of IV difference is one)	1	2^{-8}	2^{-48}	2^{-103}

Table 6. Differential characteristics for 3-initialization rounds

Input : IV	00000000000000000000000000000020	
$a_{15}^0 \cdots a_0^0$	00000000000000000000000000000000000000000000000000000000000000020	
$b_{15}^0 \cdots b_0^0$	00000000000000000000000000000000000000000000000000000000000000000	
$R1^0$	00000000000000000000000000000000	1
$R2^0$	00000000000000000000000000000000	
$R3^0$	00000000000000000000000000000000	
$a_{15}^1 \cdots a_0^1$	00000000000000000000000000000040000000000000000000000000000000000	
$b_{15}^1 \cdots b_0^1$	00000000000000000000000000000020000000000000000000000000000000000	
$R1^1$	00000000000000000000000000000020	2^{-1}
$R2^1$	00000000000000000000000000000000	
$R3^1$	00000000000000000000000000000000	
$a_{15}^2 \cdots a_0^2$	00400000000000000000000000002000000000000000000000000000000000040	
$b_{15}^2 \cdots b_0^2$	00000000002000000000000000000100000000000000000000000000000000020	
$R1^2$	00000000000000000000000000000000	2^{-8}
$R2^2$	00000000000000000000000060202040	
$R3^2$	00000000000000000000000000000000	
$a_{15}^3 \cdots a_0^3$	00000000002000000000000602020e0004000000000000000000000000000020	
$b_{15}^3 \cdots b_0^3$	00000000000000000000000000000800000000020000000000000000000010	
$R1^3$	00000020000002000000020000000	2^{-35}
$R2^3$	00000000000000000000000000000000	
$R3^3$	404080c0204060201f9d8282a06060c0	
Output : z	00000020000002000000020000008	2^{-4}

5 Bit-wise Differential Distinguisher

In this section, we first introduce single-bit and dual-bit differential cryptanalysis based on the study by Choudhuri and Maitra [3]. Then, we present an effective chosen-IV technique for our cryptanalysis of the 4-round SNOW-V. Finally, we provide the experimental results for bit-wise differential biases using the chosen-IV technique.

5.1 Single-bit and Dual-bit Differential Cryptanalysis

To find bit-wise differential biases of the reduced-round SNOW-V, we utilize single-bit and dual-bit differential cryptanalysis based on the study on the reduced-round Salsa and ChaCha, as reported by Choudhuri and Maitra [3].

Let $iv_i[j]$ be the j-th bit of the i-th element in IV for $0 \leq i \leq 7$ and $0 \leq j \leq 15$ and let $iv'_i[j]$ be an associated bit with the input difference $\Delta_{i,j}^{(0)} = iv_i[j] \oplus iv'_i[j]$,

Table 7. The differential characteristic for 4-initialization rounds

Input : IV	00000000000000000000000000000020	
$a_{15}^0 \cdots a_0^0$	0000000000000000000000000000000000000000000000000000000000000020	1
$b_{15}^0 \cdots b_0^0$	0000000000000000000000000000000000000000000000000000000000000000	
$R1^0$	00000000000000000000000000000000	
$R2^0$	00000000000000000000000000000000	
$R3^0$	00000000000000000000000000000000	
$a_{15}^1 \cdots a_0^1$	0000000000000000000000000000004000000000000000000000000000000000	2^{-1}
$b_{15}^1 \cdots b_0^1$	0000000000000000000000000000002000000000000000000000000000000000	
$R1^1$	00000000000000000000000000000020	
$R2^1$	00000000000000000000000000000000	
$R3^1$	00000000000000000000000000000000	
$a_{15}^2 \cdots a_0^2$	0040000000000000000000000000002000000000000000000000000000000040	2^{-8}
$b_{15}^2 \cdots b_0^2$	0000000000200000000000000000010000000000000000000000000000000020	
$R1^2$	00000000000000000000000000000000	
$R2^2$	00000000000000000000000060202040	
$R3^2$	00000000000000000000000000000000	
$a_{15}^3 \cdots a_0^3$	000000000020000000000000602020e0004000000000000000000000000000020	2^{-35}
$b_{15}^3 \cdots b_0^3$	0000000000000000000000000000008000000000020000000000000000000010	
$R1^3$	00000020000000200000002000000000	
$R2^3$	00000000000000000000000000000000	
$R3^3$	404080c04080c04080c0404030101020	
$a_{15}^4 \cdots a_0^4$	2060006000300020000000203010102800000000002000000000000602020e0	2^{-42}
$b_{15}^4 \cdots b_0^4$	0040000000480000000000200000004000000000000000000000000000000008	
$R1^4$	4040801000804010804040104040400 0	
$R2^4$	98818119877d7dfaa66262c400000000	
$R3^4$	00000000000000000000000000000000	
Output : z	5881010987b53dea262222d440404004	2^{-17}

which is described as $\mathcal{ID}$. Let $z_p[q]$ be the q-th bit of the p-th word in the first output keystream block z for $0 \leq p \leq 15$ and $0 \leq q \leq 7$ and let $z'_p[q]$ be an associated bit with the r-round output difference $\Delta_{p,q}^{(r)} = z_p[q] \oplus z'_p[q]$, which is described as $\mathcal{OD}$. Note that $iv_0[0]$ and $iv_7[15]$ are the least significant bit (LSB) and most significant bit (MSB) of IV, and $z_0[0]$ and $z_{15}[7]$ are the LSB and MSB of z, respectively. For a fixed key and all possible choices of IVs, single-bit and dual-bit differential probabilities are defined by

$$\Pr\big(\Delta_{p,q}^{(r)} = 1 \mid \Delta_{i,j}^{(0)} = 1\big) = \frac{1}{2}(1 + \epsilon_d), \tag{19}$$

$$\Pr\big(\Delta_{p_0,q_0}^{(r)} \oplus \Delta_{p_1,q_1}^{(r)} = 1 \mid \Delta_{i,j}^{(0)} = 1\big) = \frac{1}{2}(1 + \epsilon_d), \tag{20}$$

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

To distinguish the first keystream block z generated by the reduced-round SNOW-V from true random number sequences, we utilize the following theorem proved by Mantin and Shamir [11].

Theorem 1 ([11, Theorem 2]). *Let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions, and suppose that the event e occurs in $\mathcal{X}$ with a probability p and $\mathcal{Y}$ with a probability $p \cdot (1+q)$. Then, for small p and q, $\mathcal{O}(\frac{1}{p \cdot q^2})$ samples suffice to distinguish $\mathcal{X}$ from $\mathcal{Y}$ with a constant probability of success.*

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

5.2 Chosen-IV Technique

We analyze the source code of the `LFSR_update` algorithm in SNOW-V (refer to Listings 1-3 for details) and notice the following two properties.

Listing 1. `lfsr_update` algorithm

```
1:  typedef uint16_t u16;
2:  u16 A[16], B[16]; // The 32 cells of the two LFSRs
3:
4:  void lfsr_update ( void ){
5:      for ( int i=0; i<8; i++ ){
6:          u16 u = mul_x ( A[0], 0x990f ) ^ A[1] ^ mul_x_inv ( A[8], 0xcc87 ) ^ B[0];
7:          u16 v = mul_x ( B[0], 0xc963 ) ^ B[3] ^ mul_x_inv ( B[8], 0xe4b1 ) ^ A[0];
8:
9:          for ( int j=0; j<15; j++ ){
10:             A[j] = A[j+1];
11:             B[j] = B[j+1];
12:         }
13:
14:         A[15] = u;
15:         B[15] = v;
16:     }
17: }
```

Listing 2. `mul_x` function

```
1:  typedef uint16_t u16;
2:
3:  u16 mul_x ( u16 v, u16 c ){
4:      if ( v & 0x8000 ){
5:          return ( v << 1 ) ^ c;
6:      } else {
7:          return ( v << 1 );
8:      }
9:  }
```

Listing 3. `mul_x_inv` function

```
1:  typedef uint16_t u16;
2:
3:  u16 mul_x_inv ( u16 v, u16 d ){
4:      if ( v & 0x0001 ){
5:          return ( v >> 1 ) ^ d;
6:      } else {
7:          return ( v >> 1 );
8:      }
9:  }
```

Property 1. The `mul_x` function is executed 16 times in the `LFSR_update` algorithm, and the output varies with the value of the MSB.

Property 2. The `mul_x_inv` function is executed 16 times in the `LFSR_update` algorithm, and the output varies with the value of the LSB.

When the MSB of the input v to the mul_x function is 0, the output bits are not properly mixed because the input v is only shifted one bit to the left (see step 7 in Listing 2). On the contrary, when the MSB of the input v to the mul_x function is 1, the output bits are sufficiently mixed since the input v shifted one bit to the left is XORed with another input c (see step 5 in Listing 2). These lead to Property 1 that the MSB of the input v to the mul_x function affects whether the output bits are mixed or not. Since the mul_x_inv function is calculated in the similar manner as the mul_x function, Property 2 implies that the LSB of the input v to the mul_x_inv function affects whether the output bits are mixed or not. Furthermore, these properties may be considered to affect whether the propagation of differences is diffused or not.

Based on the two properties of the LFSR_update algorithm in SNOW-V, we present an effective chosen-IV technique for our cryptanalysis of the reduced-round SNOW-V. In the SNOW-V initialization, IV is loaded into the eight cells in the LFSR-A by assigning $(a_7, a_6, \ldots, a_0) = (iv_7, iv_6, \ldots, iv_0)$. In addition, the adversaries can choose arbitrary IVs as the $\mathcal{ID}$. Therefore, choosing IVs whose MSBs and LSBs are 0 should suppress the propagation of differences throughout the internal state of SNOW-V during the initialization phase.

We define the following eight domains for single-bit and dual-bit differential cryptanalysis.

$$\begin{aligned}
\mathcal{V}_0 &= \{\mathtt{xxxxxxxxxxxxxxx}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\},\\
\mathcal{V}_1 &= \{\mathtt{0xxxxxxxxxxxxxx0}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\},\\
\mathcal{V}_2 &= \{\mathtt{00xxxxxxxxxxxx00}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\},\\
\mathcal{V}_3 &= \{\mathtt{000xxxxxxxxx000}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\},\\
\mathcal{V}_4 &= \{\mathtt{0000xxxxxxxx0000}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\},\\
\mathcal{V}_5 &= \{\mathtt{00000xxxxxx00000}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\},\\
\mathcal{V}_6 &= \{\mathtt{000000xxxx000000}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\},\\
\mathcal{V}_7 &= \{\mathtt{0000000xx0000000}_{(2)} \mid \mathtt{x} \in \{\mathtt{0},\mathtt{1}\}\}.
\end{aligned}$$

In the next subsection, we describe our experimental observations of the bit-wise differential biases for each domain.

5.3 Experimental Results

Based on the results reported in the previous subsections, we have conducted experiments to find the bit-wise differential biases of the reduced-round SNOW-V. The following is our experimental environment: five Linux machines with 40-core Intel(R) Xeon(R) CPU E5-2660 v3 (2.60GHz), 128.0 GB of main memory, a gcc 7.2.0 compiler, and the C programming language. To find single-bit (or dual-bit) differential biases, our experiments have been conducted with 2^8 (or 2^6) trials using 2^{24} $\mathcal{ID}$s for each key, excluding domain $\mathcal{V}_7$. Since domain $\mathcal{V}_7$ contains only 2^{16} elements, we have conducted experiments with 2^{16} (or 2^{14}) trials using 2^{16} $\mathcal{ID}$s for each key to find the single-bit (or dual-bit) differential biases.

Table 8. Best single-bit and dual-bit differential biases ($\log_2$) for 4-round SNOW-V.

Domain	Single-bit			Dual-bit		
	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d
$\mathcal{V}_0$	$\Delta^{(0)}_{10,7}$	$\Delta^{(4)}_{0,0}$	-10.299	$\Delta^{(0)}_{4,1}$	$\Delta^{(4)}_{0,1} \oplus \Delta^{(4)}_{1,1}$	-9.432
$\mathcal{V}_1$	$\Delta^{(0)}_{3,1}$	$\Delta^{(4)}_{0,0}$	-10.114	$\Delta^{(0)}_{4,1}$	$\Delta^{(4)}_{0,1} \oplus \Delta^{(4)}_{1,1}$	-9.243
$\mathcal{V}_2$	$\Delta^{(0)}_{4,2}$	$\Delta^{(4)}_{0,0}$	-9.804	$\Delta^{(0)}_{4,2}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,0}$	-9.069
$\mathcal{V}_3$	$\Delta^{(0)}_{0,5}$	$\Delta^{(4)}_{2,4}$	-9.121	$\Delta^{(0)}_{4,1}$	$\Delta^{(4)}_{0,1} \oplus \Delta^{(4)}_{1,1}$	-8.825
$\mathcal{V}_4$	$\Delta^{(0)}_{6,6}$	$\Delta^{(4)}_{8,2}$	-8.975	$\Delta^{(0)}_{14,7}$	$\Delta^{(4)}_{0,1} \oplus \Delta^{(4)}_{1,1}$	-7.343
$\mathcal{V}_5$	$\Delta^{(0)}_{13,4}$	$\Delta^{(4)}_{7,3}$	-7.904	$\Delta^{(0)}_{6,7}$	$\Delta^{(4)}_{2,2} \oplus \Delta^{(4)}_{3,2}$	-5.675
$\mathcal{V}_6$	$\Delta^{(0)}_{13,1}$	$\Delta^{(4)}_{5,4}$	-6.197	$\Delta^{(0)}_{0,6}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,7}$	-3.725
$\mathcal{V}_7$	$\Delta^{(0)}_{14,1}$	$\Delta^{(4)}_{12,3}$	-4.268	$\Delta^{(0)}_{9,0}$	$\Delta^{(4)}_{0,1} \oplus \Delta^{(4)}_{3,2}$	-1.733

Table 9. Best single-bit and dual-bit differential biases ($\log_2$) for 5-round SNOW-V.

Domain	Single-bit			Dual-bit		
	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d	$\mathcal{ID}$	$\mathcal{OD}$	ϵ_d
$\mathcal{V}_0$	$\Delta^{(0)}_{12,6}$	$\Delta^{(5)}_{12,3}$	-13.943	$\Delta^{(0)}_{7,7}$	$\Delta^{(5)}_{4,2} \oplus \Delta^{(5)}_{15,5}$	-12.771
$\mathcal{V}_1$	$\Delta^{(0)}_{0,5}$	$\Delta^{(5)}_{10,6}$	-13.971	$\Delta^{(0)}_{0,1}$	$\Delta^{(5)}_{3,3} \oplus \Delta^{(5)}_{13,2}$	-12.819
$\mathcal{V}_2$	$\Delta^{(0)}_{14,7}$	$\Delta^{(5)}_{2,3}$	-14.055	$\Delta^{(0)}_{4,4}$	$\Delta^{(5)}_{4,0} \oplus \Delta^{(5)}_{14,6}$	-12.622
$\mathcal{V}_3$	$\Delta^{(0)}_{4,0}$	$\Delta^{(5)}_{15,1}$	-14.021	$\Delta^{(0)}_{11,0}$	$\Delta^{(5)}_{9,3} \oplus \Delta^{(5)}_{11,7}$	-12.671
$\mathcal{V}_4$	$\Delta^{(0)}_{1,4}$	$\Delta^{(5)}_{5,2}$	-14.147	$\Delta^{(0)}_{9,1}$	$\Delta^{(5)}_{8,3} \oplus \Delta^{(5)}_{14,3}$	-12.713
$\mathcal{V}_5$	$\Delta^{(0)}_{15,4}$	$\Delta^{(5)}_{2,0}$	-14.047	$\Delta^{(0)}_{6,7}$	$\Delta^{(5)}_{7,5} \oplus \Delta^{(5)}_{15,4}$	-12.669
$\mathcal{V}_6$	$\Delta^{(0)}_{2,5}$	$\Delta^{(5)}_{15,6}$	-14.081	$\Delta^{(0)}_{11,1}$	$\Delta^{(5)}_{4,0} \oplus \Delta^{(5)}_{15,2}$	-12.820
$\mathcal{V}_7$	$\Delta^{(0)}_{6,7}$	$\Delta^{(5)}_{6,7}$	-13.589	$\Delta^{(0)}_{0,7}$	$\Delta^{(5)}_{1,2} \oplus \Delta^{(5)}_{6,1}$	-12.408

Tables 8 and 9 show the best single-bit and dual-bit differential biases for the four and five rounds of SNOW-V. As shown in Table 8, we obtain higher biases when the domain is restricted using the chosen-IV technique. For example, we obtain the best single-bit (or dual-bit) differential bias of $\epsilon_d = 2^{-4.268}$ (or $2^{-1.733}$) for domain $\mathcal{V}_7$, whereas we find $\epsilon_d = 2^{-10.299}$ (or $2^{-9.432}$) for domain $\mathcal{V}_0$. However, as shown in Table 9, all of the best single-bit and dual-bit differential biases are almost constant regardless of the domain in the 5-round SNOW-V. These results demonstrate that the chosen-IV technique is valid for the 4-round SNOW-V, but not for the 5-round SNOW-V.

For the 4-round SNOW-V, the best dual-bit differential bias in domain $\mathcal{V}_7$, i.e., $\epsilon_d = 2^{-1.733}$, provides a practical differential distinguisher. According to Theorem 1, $2^{4.466}$ samples suffice to distinguish the 4-round SNOW-V from a true random number generator with a constant probability of success. Similarly, for the 5-round SNOW-V, the best dual-bit differential bias in domain $\mathcal{V}_2$, i.e., $\epsilon_d = 2^{-12.622}$, provides the best differential distinguisher. Although the best dual-bit differential bias in domain $\mathcal{V}_7$ is higher than that in $\mathcal{V}_2$, i.e., $\epsilon_d = 2^{-12.408}$, that in domain $\mathcal{V}_7$ cannot provide the best differential distinguisher because domain $\mathcal{V}_7$ contains only 2^{16} elements. Thus, $2^{26.244}$ samples suffice to distinguish

the 5-round SNOW-V from a true random number generator; however, the accuracy of the experimental results may be insufficient because we have conducted experiments with only 2^{24} $\mathcal{ID}$s to observe the differential biases. To find more precise dual-bit differential biases for the 5-round SNOW-V, we have focused on the best $\mathcal{ID}$-$\mathcal{OD}$ pair in each domain (excluding domain $\mathcal{V}_7$) listed in Table 9, and have conducted additional experiments with 2^8 trials using 2^{32} $\mathcal{ID}$s for each key. Consequently, we obtain the best dual-bit differential biases in domain $\mathcal{V}_4$, such that $\mathcal{ID}$ is $\Delta^{(0)}_{9,1}$, $\mathcal{OD}$ is $\Delta^{(5)}_{8,3} \oplus \Delta^{(5)}_{14,3}$, and ϵ_d is approximately $2^{-17.934}$. Thus, our experiments have revealed that at least $2^{36.868}$ samples suffice to distinguish the 5-round SNOW-V from a true random number generator.

6 Key Recovery Attack on the 4-round SNOW-V

In this section, we describe a key recovery attack on the 4-round SNOW-V. To the best of our knowledge, our attack is the best key recovery attack on the reduced-round SNOW-V since the cube attack on the 3-round SNOW-V proposed by Ekdahl et al. [5], which was the best to date. Our proposed attack is an improvement on the differential attack based on a technique called *probabilistic neutral bits* (PNB) proposed by Aumasson et al. [2].

6.1 Differential Attack Based on Probabilistic Neutral Bits (PNB)

Aummason et al. proposed a differential attack based on PNB and applied it to Salsa and ChaCha [2]. In this subsection, we introduce their attack to clarify the difference from our proposed attack, which is described in Section 6.2. Their attack consists of two phases: precomputation and online phases. The precomputation phase is further divided into three phases: differential characteristic search (as described in Section 5.1), PNB identification, and probabilistic backwards computation phases.

PNB Identification Phase. PNB is a concept which divides the secret key bits into two sets: m-bit significant key bits and n-bit non-significant key bits. To identify these two sets, Aumasson et al. focused on the amount of influence which each secret key bit has on the output difference $\mathcal{OD}$, and defined that amount as *neutral measure*.

Definition 1 ([2, Definition 1]). *The neutral measure of the key bit κ_i with respect to the output difference $\mathcal{OD}$ is defined as γ_i, where $\Pr = \frac{1}{2}(1+\gamma_i)$ is the probability that complementing the key bit κ_i does not change the $\mathcal{OD}$.*

For example, according to Definition 1, we have the following singular cases of the neutral measure:

- $\gamma_i = 1$: $\mathcal{OD}$ does not depend on the i-th key bit, i.e., it is non-significant.
- $\gamma_i = 0$: $\mathcal{OD}$ is statistically independent of the i-th key bit, i.e., it is significant.

To identify the PNB by using the concept of the neutral measure, we perform the following procedure after the differential characteristic search phase:

Step 1. Compute the keystream pair Z, Z' corresponding to the input pair $X^{(0)}, X'^{(0)}$ with the input difference $\Delta_{i,j}^{(0)}$. Note that the keystream Z is derived by $X^{(0)} + X^{(R)}$ in the case of Salsa and ChaCha.

Step 2. Prepare a new input pair $\overline{X}^{(0)}, \overline{X'}^{(0)}$ with the key bit position i of the original input pair $X^{(0)}, X'^{(0)}$ flipped by one bit.

Step 3. Compute the internal state pair $Y^{(r)}, Y'^{(r)}$ with $Z - \overline{X}^{(0)}$, $Z' - \overline{X'}^{(0)}$ for $r < R$, as inputs to the inverse function of the initialization in the case of Salsa and ChaCha.

Step 4. Compute $\Gamma_{p,q}^{(r)} = y_p[q] \oplus y'_p[q]$, where $y_p[q]$ and $y'_p[q]$ are the q-th bit of the p-th word of $Y^{(r)}$ and $Y'^{(r)}$, respectively.

Step 5. Repeatedly perform Steps 1-4 by using different input pairs with the same $\Delta_{i,j}^{(0)}$; compute the neutral measure as $\Pr(\Delta_{p,q}^{(r)} = \Gamma_{p,q}^{(r)} \mid \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \gamma_i)$, where $\Delta_{p,q}^{(r)}$ is the output difference derived during the differential characteristic search (as described in Section 5.1).

Step 6. Set a threshold γ and put all key bits with $\gamma_i < \gamma$ into a set of significant key bits (of size m) and those with $\gamma_i \geq \gamma$ into a set of non-significant key bits (of size n).

Probabilistic Backwards Computation Phase. In the differential characteristic search phase, we derive the r-th round differential biases from input pairs with the chosen input difference, i.e., this implies that we perform the forwards computation in the target cipher. However, in the case of Salsa and ChaCha, we can also derive the r-th round differential biases from the obtained keystream by performing the backwards computation, which is called the *probabilistic backwards computation*.

In the probabilistic backwards computation phase, we perform the following procedure after the PNB identification phase:

Step 1. Compute the keystream pair Z, Z' corresponding to the input pair $X^{(0)}, X'^{(0)}$ with the input difference $\Delta_{i,j}^{(0)}$.

Step 2. Prepare a new input pair $\hat{X}^{(0)}, \hat{X}'^{(0)}$ with only non-significant key bits reset to a fixed value (e.g., all zero) from the original input pair $X^{(0)}, X'^{(0)}$.

Step 3. Compute the internal state pair $\hat{Y}^{(r)}, \hat{Y}'^{(r)}$ with $Z - \hat{X}^{(0)}$, $Z' - \hat{X}'^{(0)}$ for $r < R$, as inputs to the inverse function of the initialization in the case of Salsa and ChaCha.

Step 4. Compute $\hat{\Gamma}_{p,q}^{(r)} = \hat{y}_p[q] \oplus \hat{y}'_p[q]$, where $\hat{y}_p[q]$ and $\hat{y}'_p[q]$ are the q-th bit of the p-th word of $\hat{Y}^{(r)}$ and $\hat{Y}'^{(r)}$, respectively.

Step 5. Repeatedly perform Steps 1-4 by using different input pairs with the same $\Delta_{i,j}^{(0)}$; compute the r-round bias ϵ_a as $\Pr(\Delta_{p,q}^{(r)} = \hat{\Gamma}_{p,q}^{(r)} \mid \Delta_{i,j}^{(0)} = 1) = \frac{1}{2}(1 + \epsilon_a)$, where $\Delta_{p,q}^{(r)}$ is the output difference derived during the differential characteristic search (as described in Section 5.1).

According to [2], the bias ϵ is approximated as $\epsilon_d \cdot \epsilon_a$ and considered to compute the overall complexity of the attack on the R-round target cipher.

Online Phase. According to [2], we perform the following procedure after the precomputation phase:

Step 1. For an unknown key, we collect N keystream pairs where each pair is generated by a random input pair (satisfying the relevant input difference).

Step 2. For each choice of the subkey (i.e., the m-bit significant key bits) do:

- **Step 2-1.** Derive the r-th round differential biases from the N keystream pairs by performing the backwards computation.
- **Step 2-2.** If the optimal distinguisher legitimates the subkeys candidate as a (possibly) correct one, we perform an additional exhaustive search over the n non-significant key bits in order to check the correctness of this filtered subkey and to find the non-significant key bits.
- **Step 2-3.** Stop if the correct key is found, and output the recovered key.

Complexity Estimation. According to [2, 3], given samples N and probability of false alarm is $P_{fa} = 2^{-\alpha}$, the time complexity of the attack is given by

$$2^m(N + 2^n P_{fa}) = 2^m N + 2^{256-\alpha}, \text{ where } N \approx \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1-\epsilon^2}}{\epsilon}\right)^2,$$

for probability of non-detection $P_{nd} = 1.3 \times 10^{-3}$. In practice, α (and hence N) is chosen such that it minimizes the time complexity of the attack.

6.2 Application to SNOW-V

In this subsection, we present how to apply the differential attack based on PNB, as described in Section 6.1, to the reduced-round SNOW-V. However, its application to SNOW-V, unlike the existing attacks on Salsa and ChaCha, is difficult to compute the difference biases from the obtained keystreams by performing the backwards computation, i.e., it is difficult to perform in the same procedure as Step 3 in the PNB identification phase and Step 3 in the probabilistic backwards computation phase, as described in Section 6.1.

To solve this problem, in our proposed attack, we replace the backwards computations in these steps with the forwards computations. Our attack consists of three precomputation phases: differential characteristic search (as described in Section 5.1), PNB identification, and probabilistic *forwards* computation phases. The online phase is a similar procedure to that described in Section 6.1, i.e., we simply replace the backwards computation with the forwards computation in Step 2-1 of the online phase.

PNB Identification Phase. In the PNB identification phase, we replace Step 3 in the existing phase with a step to perform the forwards computation. Additionally, it is not necessary to perform Step 1 in the existing phase because no backwards computation is performed. In summary, for the application to SNOW-V, we perform the following procedure after the differential characteristic search phase:

Step 1. Prepare a new input pair $\overline{X}^{(0)}, \overline{X'}^{(0)}$ with the key bit position i of the original input pair $X^{(0)}, X'^{(0)}$ flipped by one bit. Note that, according to Section 2.2, an input $X^{(0)}$ of SNOW-V is initialized from a secret key and an initialization vector.

Step 2. Compute the keystream pair z, z' with $\overline{X}^{(0)}$, $\overline{X'}^{(0)}$ as inputs to the r-round initialization of SNOW-V.

Step 3. Compute $\Gamma^{(r)}_{p,q} = z_p[q] \oplus z'_p[q]$, where $z_p[q]$ and $z'_p[q]$ are the q-th bit of the p-th word of z and z', respectively.

Step 4. Repeatedly perform Steps 1-4 by using different input pairs with the same $\Delta^{(0)}_{i,j}$; compute the neutral measure as $\Pr(\Delta^{(r)}_{p,q} = \Gamma^{(r)}_{p,q} \mid \Delta^{(0)}_{i,j} = 1) = \frac{1}{2}(1+\gamma_i)$, where $\Delta^{(r)}_{p,q}$ is the output difference derived during the differential characteristic search (as described in Section 5.1).

Step 5. Set a threshold γ, put all key bits with $\gamma_i < \gamma$ into a set of significant key bits (of size m) and those with $\gamma_i \geq \gamma$ into a set of non-significant key bits (of size n).

Probabilistic Forwards Computation Phase. Similar to the proposed PNB identification phase, we improve the existing probabilistic backwards computation phase. In summary, for the application to SNOW-V, we perform the following procedure after the PNB identification phase:

Step 1. Prepare a new input pair $\hat{X}^{(0)}, \hat{X}'^{(0)}$ with only non-significant key bits reset to a fixed value (e.g., all zero) from the original input pair $X^{(0)}, X'^{(0)}$.

Step 2. Compute the keystream pair $\hat{z}, \hat{z}'$ with $\hat{X}^{(0)}$, $\hat{X}'^{(0)}$ as inputs to the r-round initialization of SNOW-V.

Step 3. Compute $\hat{\Gamma}^{(r)}_{p,q} = \hat{z}_p[q] \oplus \hat{z}'_p[q]$, where $\hat{z}_p[q]$ and $\hat{z}'_p[q]$ are the q-th bit of the p-th word of $\hat{z}^{(r)}$ and $\hat{z}'^{(r)}$, respectively.

Step 4. Repeatedly perform Steps 1-4 by using different input pairs with the same $\Delta^{(0)}_{i,j}$; compute the r-round bias ϵ_a as $\Pr(\Delta^{(r)}_{p,q} = \hat{\Gamma}^{(r)}_{p,q} \mid \Delta^{(0)}_{i,j} = 1) = \frac{1}{2}(1+\epsilon_a)$, where $\Delta^{(r)}_{p,q}$ is the output difference derived during the differential characteristic search (as described in Section 5.1).

Complexity Estimation. In our proposed attack, we can construct the following two independent distinguishers:

- A distinguisher based on the differential bias ϵ_d.
- A distinguisher based on the bias ϵ_a.

This is because these biases are derived from the (secret) internal states in the existing attacks, whereas they are derived from the keystreams, which are obtained by an adversary under the known plaintext attack scenario, in the application to SNOW-V. Thus, the number of samples N for our attack is given by

$$N \approx \max\left(\left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1-\epsilon_d^2}}{\epsilon_d}\right)^2, \left(\frac{\sqrt{\alpha \log 4} + 3\sqrt{1-\epsilon_a^2}}{\epsilon_a}\right)^2\right).$$

Additionally, the time complexity of our attack is given in the same way as that of the existing attacks [2, 3], as described in Section 6.1.

6.3 Experimental Results

Based on the attack procedure proposed in the previous subsection, we have conducted experiments to find the best parameters for our attack on the 4-round SNOW-V. The following is our experimental environment: five Linux machines with 40-core Intel(R) Xeon(R) CPU E5-2660 v3 (2.60GHz), 128.0 GB of main memory, a gcc 7.2.0 compiler, and the C programming language. To find the best parameters for our attack, our experiments have been conducted with 2^8 trials using 2^{24} $\mathcal{ID}$s for each key excluding domain $\mathcal{V}_7$. Since domain $\mathcal{V}_7$ contains only 2^{16} elements, we have conducted experiments with 2^{16} trials using 2^{16} $\mathcal{ID}$s for each key. In addition, we need to consider the possibility that our attack has no validity because the application to SNOW-V, unlike the existing attacks on Salsa and ChaCha, only perform the forwards computation throughout all phases. To calculate the success probability of our attack, our experiments have been conducted with 1000 trials by using the best parameters obtained from the experiments. In our experiments, we consider the attack to be failed if we can guess a subkey candidate with a higher bias ϵ_a^* than the bias ϵ_a obtained from the correctly guessed subkey.

Tables 10 and 11 show the best parameters for our attack in domains $\mathcal{V}_0$ and $\mathcal{V}_7$ on the 4-round SNOW-V for each threshold γ. Based on these tables, we appear to be able to perform our attack on the 4-round SNOW-V with the least time complexity of $2^{60.35}$ by using the parameter for the threshold $\gamma = 0.50$ in domain $\mathcal{V}_7$, but it has no validity because its success probability is zero. However, as shown in these tables, we can perform our attack with a success probability of one by using the parameter for the threshold $\gamma = 1.00$ in both domains $\mathcal{V}_0$ and $\mathcal{V}_7$. This is because all key bits with a threshold $\gamma_i \geq \gamma = 1.00$ are put into the set of non-significant key bits, and these have no influence on the output difference, i.e., this implies that we can always guess all the m-bits subkeys in the online phase. As a result, we can perform our attack on the 4-round SNOW-V with a time complexity of $2^{153.97}$ and data complexity of $2^{26.96}$ by using the parameter for the threshold $\gamma = 1.00$ in domain $\mathcal{V}_0$; this is the best key recovery attack on the reduced-round SNOW-V.

Table 10. The best parameters for our attack in domain $\mathcal{V}_0$ for the 4-round SNOW-V for each threshold γ, where m is the size of significant key bits.

γ	$\mathcal{ID}$	$\mathcal{OD}$	m	ϵ_a	ϵ_d	α	Data	Time	Probability
1.00	$\Delta^{(0)}_{4,3}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,0}$	127	1.000	$2^{-9.548}$	109	$2^{26.96}$	$2^{153.97}$	1.000
0.90	$\Delta^{(0)}_{4,0}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,0}$	82	$2^{-0.183}$	$2^{-9.546}$	154	$2^{27.36}$	$2^{109.37}$	0.958
0.80	$\Delta^{(0)}_{4,0}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,0}$	72	$2^{-0.538}$	$2^{-9.546}$	164	$2^{27.44}$	$2^{99.45}$	0.729
0.70	$\Delta^{(0)}_{4,0}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,0}$	67	$2^{-0.714}$	$2^{-9.546}$	169	$2^{27.48}$	$2^{94.48}$	0.650
0.60	$\Delta^{(0)}_{4,0}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,0}$	66	$2^{-0.830}$	$2^{-9.546}$	170	$2^{27.48}$	$2^{93.49}$	0.542
0.50	$\Delta^{(0)}_{4,3}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,0}$	61	$2^{-1.388}$	$2^{-9.548}$	175	$2^{27.52}$	$2^{88.53}$	0.334

Table 11. The best parameters for our attack in domain $\mathcal{V}_7$ for the 4-round SNOW-V for each threshold γ, where m is the size of significant key bits.

γ	$\mathcal{ID}$	$\mathcal{OD}$	m	ϵ_a	ϵ_d	α	Data	Time	Probability
1.00	$\Delta^{(0)}_{1,6}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,1}$	149	1.000	$2^{-1.878}$	108	$2^{11.59}$	$2^{154.60}$	1.000
0.90	$\Delta^{(0)}_{10,7}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{2,7}$	84	$2^{-0.123}$	$2^{-1.747}$	167	$2^{11.84}$	$2^{95.86}$	0.858
0.80	$\Delta^{(0)}_{10,7}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{2,7}$	74	$2^{-0.570}$	$2^{-1.747}$	177	$2^{11.91}$	$2^{85.93}$	0.253
0.70	$\Delta^{(0)}_{10,7}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{2,7}$	71	$2^{-0.646}$	$2^{-1.747}$	181	$2^{11.94}$	$2^{81.95}$	0.150
0.60	$\Delta^{(0)}_{10,7}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{2,7}$	64	$2^{-1.037}$	$2^{-1.747}$	187	$2^{11.98}$	$2^{75.99}$	0.012
0.50	$\Delta^{(0)}_{1,6}$	$\Delta^{(4)}_{0,0} \oplus \Delta^{(4)}_{1,1}$	47	$2^{-1.742}$	$2^{-1.878}$	203	$2^{12.35}$	$2^{60.35}$	0.000

7 Conclusion

In this study, we analyzed the security of SNOW-V with three attacks: the MILP-aided integral attack, the MILP-aided differential attack, and the bit-wise differential bias attack. These attacks allow us to construct distinguishers of up to five rounds. Furthermore, the differential biases obtained by the bit-wise differential bias attack can be integrated into our improved key recovery attack based on probabilistic neutral bits, which is inspired by the existing study on Salsa and ChaCha [2, 3]. As a result, we present the best key recovery attack on the 4-round version with a time complexity of $2^{153.97}$ and data complexity of $2^{26.96}$. Consequently, we have improved the best existing attack, which was evaluated by the designers, in the initialization phase of the reduced-round SNOW-V.

Acknowledgments Takanori Isobe is supported by JST, PRESTO Grant Number JPMJPR2031 and SECOM science and technology foundation.

References

[1] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP modeling for (large) s-boxes to optimize probability of differential characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017.

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

[3] Arka Rai Choudhuri and Subhamoy Maitra. Significantly improved multi-bit differentials for reduced round Salsa and ChaCha. *IACR Trans. Symm. Cryptol.*, 2016(2):261–287, 2016.

[4] CNET. Logic friday. `https://download.cnet.com/Logic-Friday/3000-20415_4-75848245.html/`.

[5] Patrik Ekdahl, Thomas Johansson, Alexander Maximov, and Jing Yang. A new SNOW stream cipher called SNOW-V. *IACR Trans. Symmetric Cryptol.*, 2019(3):1–42, 2019.

[6] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016*, volume 9783 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2016.

[7] Yuki Funabiki, Yosuke Todo, Takanori Isobe, and Masakatu Morii. Several MILP-Aided Attacks Against SNOW 2.0. In Jan Camenisch and Panos Papadimitratos, editors, *Cryptology and Network Security - 17th International Conference, CANS 2018*, volume 11124 of *Lecture Notes in Computer Science*, pages 394–413. Springer, 2018.

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

[9] Lin Jiao, Yongqiang Li, and Yonglin Hao. A Guess-And-Determine Attack On SNOW-V Stream Cipher. *The Computer Journal*, 2020.

[10] Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2001.

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

[12] Bing Sun, Zhiqiang Liu, Vincent Rijmen, Ruilin Li, Lei Cheng, Qingju Wang, Hoda AlKhzaimi, and Chao Li. Links Among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference*, volume 9215 of *Lecture Notes in Computer Science*, pages 95–115. Springer, 2015.

[13] Ling Sun, Wei Wang, and Meiqin Wang. MILP-Aided Bit-Based Division Property for Primitives with Non-Bit-Permutation Linear Layers. *IACR Cryptol. ePrint Arch.*, 2016:811, 2016.

[14] Ling Sun, Wei Wang, and Meiqin Wang. Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on teh Theory and Applications of Cryptology and Information Security*, volume 10624 of *Lecture Notes in Computer Science*, pages 128–157. Springer, 2017.

[15] Yosuke Todo. Structural Evaluation by Generalized Integral Property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on teh Theory and Applications of*

Cryptographic Techniques, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015.

[16] Yosuke Todo and Masakatu Morii. Bit-Based Division Property and Application to Simon Family. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016*, volume 9783 of *Lecture Notes in Computer Science*, pages 357–377. Springer, 2016.

[17] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on teh Theory and Application of Cryptology and Information Security*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.