IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2021, No. 1, pp. 378–410. DOI:10.46586/tosc.v2021.i1.378-410

Resistance of SNOW-V against Fast Correlation Attacks

Xinxin Gong[1] and Bin Zhang[2,1,3,4]

[1] State Key Laboratory of Cryptology, P. O. Box 5159, Beijing, 100878, China,
xinxgong@126.com, martin_zhangbin@hotmail.com
[2] TCA Laboratory, SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing, China,
[3] University of Chinese Academy of Sciences, Beijing, 100049, China,
[4] Guizhou shujubao Network Technology Co., Ltd, Guizhou, China

Abstract. SNOW-V is a new member in the SNOW family of stream ciphers, hoping to be competitive in the 5G mobile communication system. In this paper, we study the resistance of SNOW-V against *bitwise* fast correlation attacks by constructing bitwise linear approximations. First, we propose and summarize some efficient algorithms using the slice-like techniques to compute the bitwise linear approximations of certain types of composition functions composed of basic operations like $\boxplus$, $\oplus$, *Permutation*, and *S-box*, which have been widely used in word-oriented stream ciphers such as SNOW-like ciphers. Then, using these algorithms, we find a number of stronger linear approximations for the FSM of the two variants of SNOW-V given in the design document, i.e., SNOW-V$_{\sigma_0}$ and SNOW-V$_{\boxplus_8,\boxplus_8}$. For SNOW-V$_{\sigma_0}$, where there is no byte-wise permutation, we find some bitwise linear approximations of the FSM with the SEI (Squared Euclidean Imbalance) around $2^{-37.34}$ and mount a bitwise fast correlation attack with the time complexity $2^{251.93}$ and memory complexity 2^{244}, given $2^{103.83}$ keystream outputs, which improves greatly the results in the design document. For SNOW-V$_{\boxplus_8,\boxplus_8}$, where both of the two 32-bit adders in the FSM are replaced by 8-bit adders, we find our best bitwise linear approximations of the FSM with the SEI $2^{-174.14}$, while the best byte-wise linear approximation in the design document of SNOW-V has the SEI $2^{-214.80}$. Finally, we study the security of a closer variant of SNOW-V, denoted by SNOW-V$_{\boxplus_{32},\boxplus_8}$, where only the 32-bit adder used for updating the first register is replaced by the 8-bit adder, while everything else remains identical. For SNOW-V$_{\boxplus_{32},\boxplus_8}$, we derive many mask tuples yielding the bitwise linear approximations of the FSM with the SEI larger than 2^{-184}. Using these linear approximations, we mount a fast correlation attack with the time complexity $2^{377.01}$ and a memory complexity 2^{363}, given $2^{253.73}$ keystream outputs. Note that neither of our attack threatens the security of SNOW-V. We hope our research could further help in understanding bitwise linear approximation attacks and also the structure of SNOW-like stream ciphers.

Keywords: Stream ciphers · SNOW-V · FSM · Bitwise Fast correlation attack · Byte-wise Linear Approximations

1 Introduction

1.1 Background

SNOW-V [8] is a new member in the SNOW family of stream ciphers, hoping to be competitive in the 5G mobile communication system. It is designed by revising the SNOW 3G architecture and has kept the general design from SNOW 3G. SNOW 3G [9] is one member of the SNOW family with two predecessors SNOW 1.0 [7] and SNOW 2.0 [6].

Received: 2020-11-23 Accepted: 2021-01-23 Published: 2021-03-19

SNOW 1.0 was submitted to NESSIE project by Ekdahl and Johansson in 2000, and SNOW 2.0 is an improved version which was published in 2002 and selected as an ISO standard in 2005. Both SNOW 1.0 and SNOW 2.0 consist of two main components: a Linear Feedback Shift Register (LFSR) and a Finite State Machine (FSM), based on operations on 32-bit words, with high efficiency in both software and hardware environment. SNOW 3G was designed in 2006 by ETSI/SAGE, different from SNOW 2.0 by introducing a third 32-bit register to the FSM and a corresponding 32-bit nonlinear transformation for updating this register. SNOW 3G serves as the core of 3GPP Confidentiality and Integrity Algorithms UEA 2 & UIA2 for UMTS and LTE networks. It is currently in use in 3-4G mobile telephony systems, while SNOW-V aims to adapt SNOW 3G for 5G.

SNOW-V has kept most of the design from SNOW 3G in terms of the LFSR and the FSM, but both components are updated to better align with vectorized implementations. The LFSR part is now a circular construction consisting of two LFSRs, each feeding into the other, and the size of each register in the FSM part has been increased from 32 bits to 128 bits. At each clock, SNOW-V generates a 128-bit keystream. The original version of SNOW-V appeared on the IACR ePrint on November 29, 2018, and later a stronger version was posted, where a byte-wise permutation σ was added in the updating function of the first register $R1$ of the FSM.

Linear approximation attacks, including distinguishing attacks and correlation attacks, have been widely used to analyze SNOW ciphers. The basic technique is to approximate the nonlinear operations in the cipher and then derive a linear approximation relation involving the keystream symbols. If the linear approximation involves also symbols from the LFSR states, a correlation attack can be mounted by utilizing some correlation between the keystream and the LFSR states. We give the references [4, 5] for the basic foundations of correlation attacks.

1.2 Related Work

The resistance of SNOW 2.0 against distinguishing attacks and correlation attacks has been widely studied. In these attacks, the first step is to approximate the FSM part through the linear masking method as proposed in [3], and then to cancel out the contributions of the registers by combining the expressions for several keystream words at different time instances. In [20] and [19], distinguishing attacks were given with the complexities 2^{225} and 2^{174} respectively. At Asiacrypt 2008, a correlation attack [15] was proposed with the complexity $2^{212.38}$ by building the bitwise linear approximations for the FSM. Note that all the attacks in [20, 19, 15] were based on the bitwise linear approximations. At CRYPTO 2015, Zhang et al. [23] introduced the terminology "large-unit" linear approximations, and mounted a fast correlation attack on SNOW 2.0 by building the byte-wise (8-bit) linear approximations, giving the significantly reduced complexities all below $2^{164.15}$. In this process, they derived two types of byte-wise linear approximations, and accordingly provided two algorithms to compute the bias using the Squared Euclidean Imbalance (SEI) as defined in [1] with the complexities $2^{33.58}$ and $2^{26.58}$ respectively for each given byte-wise mask tuple. Later in [10], the correlation attack on SNOW 2.0 was improved slightly with the complexity $2^{162.91}$. Recently, [11] investigated the bitwise linear approximation of a certain type of composition function present in SNOW 2.0 and proposed a linear-time algorithm to compute the correlation for an arbitrary given linear mask. Based on this algorithm, they carried out a wider range of search for bitwise masks and found some strong linear approximations which enable them to slightly improve the data complexity of the previous fast correlation attacks by using multiple bitwise linear approximations.

For SNOW 3G, the bitwise linear approximations over three rounds of the FSM were depicted in [19], but only rough estimates of the upper bounds of their correlations were given. In [11], a fast correlation attack was given by constructing the bitwise linear approximations whose correlations were accurately computed. In [22], inspired by the

results of [23] where the large-unit approach was used to achieve improvements over the previous attacks on SNOW 2.0, Yang et al. constructed the three-round byte-wise linear approximations for the FSM of SNOW 3G and performed the searches for finding actual byte-wise masks that gave high SEI values for the approximations. The byte-wise linear approximations found in [22] were also applied to launch a fast correlation attack against SNOW 3G.

For SNOW-V, there is no prior cryptanalysis of SNOW-V beyond its design document [8], except for a result published in December 2020 [12] where a byte-based guess and determine attack was proposed with complexity 2^{406}. In [8], the designers presented their best results of the linear approximation attacks on several close variants of SNOW-V, including the original version of SNOW-V denoted by SNOW-V$_{\sigma_0}$, where the permutation is assumed to be just the identity σ_0, and also SNOW-V$_{\boxplus_8,\boxplus_8}$, where both of the two 32-bit adders "$\boxplus_{32}$" in the updating function of the FSM are replaced by the 8-bit ones "$\boxplus_8$" while everything else remains identical. In all these analyses, they utilized "large-unit" linear approximations. For SNOW-V$_{\sigma_0}$, they constructed the byte-wise linear approximations and computed the bias using the SEI, giving their best result with the SEI $2^{-58.7}$. With this byte-wise linear approximation, they also mounted a fast correlation attack following the method in [23] with the time complexity of about 2^{232}, requiring a keystream of length 2^{203}. In the course of computing the SEI for any given byte-wise mask, the convolution algorithms were used to compute large distributions which were computationally demanding. Beside, they also searched for the byte-wise linear approximations for the FSM of SNOW-V$_{\boxplus_8,\boxplus_8}$. In their best attempt, they got the total noise having the SEI $2^{-214.80}$. To the best of our knowledge, there have been no significant research on SNOW-V in published literature until now.

1.3 Our Contributions

In this paper, we investigate the bitwise linear approximations for the FSM of SNOW-V through linear masking, and present fast correlation attacks on several close variants.

First, we summarize five types of sub-functions composed of basic operations like $\boxplus$, $\oplus$, *Permutation*, and *S-box*, which have been widely used in word-oriented stream ciphers such as SNOW-like stream ciphers, and propose some linear-time algorithms to compute the correlation of the bitwise linear approximation for an arbitrary given linear mask tuple. For Type-I to Type-III, we utilize linear-time algorithms from [11, 19], and for Type-IV and Type-V, we propose new linear-time algorithms for efficiently computing the bitwise linear approximations by extending the techniques in [11, 19]. All these algorithms use a technique we call "slice-like" to efficiently compute the correlations. The general idea of the "slice-like" technique is to divide the n-bit values into d m-bit values ($n = md$) according to the specific structure of the underlying function (m-bit S-box, for example), and then pre-compute and store some specific matrices independent of the given linear mask, and finally compute the correlation for any given mask in linear-time using these pre-computed matrices. The novelty is the construction of specific matrices which can be efficiently pre-computed. Using the slice-like techniques, the computations of bitwise linear approximations cost only linear-time complexities by doing some matrix multiplications, while the convolution algorithms on large distribution in [8] need much more computations. Based on these algorithms, we are able to search for the bitwise linear masks in a much larger range than the designers do for the byte-wise masks [8].

Then we apply these algorithms for the cryptanalysis of SNOW-V against linear approximation attacks. Our attacks target three variants of SNOW-V, two were introduced by SNOW-V designers in [8], while the third one is new.

- For SNOW-V$_{\sigma_0}$, we find a number of bitwise linear approximations which have significantly larger SEI values than that of the best byte-wise linear approximation

found in [8], and present a fast correlation attack by using these new-found bitwise linear approximations. This attack costs a time complexity of $2^{251.93}$ and a memory complexity of 2^{244}, less than the exhaustive key search, and requires a keystream of length around $2^{103.83}$, which improves greatly the result in [8] which is 2^{203}.

- For SNOW-V$_{\boxplus_8,\boxplus_8}$, we give a brief study on the bitwise linear approximation of the FSM. In our attempt to approximate the FSM, we have found our best bitwise linear approximation with the SEI $2^{-174.14}$, while the SEI of the best byte-wise linear approximation found in [8] is $2^{-214.80}$.
- For SNOW-V$_{\boxplus_{32},\boxplus_8}$, a new and closer variant which has the byte-wise permutation σ but only the 32-bit adder used for updating the first register is replaced by the 8-bit adder, we derive many bitwise mask tuples yielding the bitwise linear approximations with the SEI larger than 2^{-184}. Using these linear approximations, we mount a fast correlation attack with the time complexity $2^{377.01}$ and a memory complexity 2^{363}, given $2^{253.73}$ keystream outputs.

Note that neither of our attacks threatens the security of SNOW-V. But we hope our research could further help in understanding bitwise linear approximation attacks and also the structure of SNOW-like stream ciphers.

1.4 Paper Organization

Some basic notations and definitions are presented in Section 2 together with a brief description of SNOW-V. In Section 3, we propose and summarize some algorithms to efficiently compute the bitwise linear approximations of certain types of composition functions. In Section 4 and Section 5, we apply these algorithms to the bitwise linear approximations of the FSM of SNOW-V$_{\sigma_0}$ and SNOW-V$_{\boxplus_{32},\boxplus_8}$, respectively. A brief study on the bitwise linear approximation of the FSM of SNOW-V$_{\boxplus_8,\boxplus_8}$ is given in Section 6. Finally, some conclusions are provided with the future work pointed out in Section 7.

2 Preliminaries

2.1 Notations and Definitions

The following notations and definitions are used throughout this paper.

- The bitwise exclusive-OR is denoted by "$\oplus$" and the addition modulo 2^m by "$\boxplus_m$".
- The binary field is denoted by $\mathbf{F}_2$ and its m-dimensional extension field is denoted by $\mathbf{F}_{2^m}$. Besides, we denote by $\mathbf{F}_{2^m}^*$ the multiplicative group of nonzero elements of $\mathbf{F}_{2^m}$.
- Given two binary vectors $\mathbf{a} = (a_0, a_1, ..., a_{m-1}) \in \mathbf{F}_{2^m}$ and $\mathbf{b} = (b_0, b_1, ..., b_{m-1}) \in \mathbf{F}_{2^m}$, the standard inner product is defined as $\mathbf{a} \cdot \mathbf{b} = \bigoplus_{i=0}^{m-1} a_i b_i$, where the product $a_i b_i$ is taken in $\mathbf{F}_2$.
- Let n, m be two positive integers such that m divides n and $d = \frac{n}{m}$. For $\mathbf{x} \in \mathbf{F}_{2^n}$, it can be written as $\mathbf{x} = (\mathbf{x}_0 \parallel ... \parallel \mathbf{x}_{d-1})$, where $\mathbf{x}_i \in \mathbf{F}_{2^m}$ for $0 \leq i \leq d-1$, and $\mathbf{x}_0$ is the least significant part.
- For a set S, the number of elements in S is denoted by $|S|$.
- Let X be a binary random variable, the correlation between X and zero is defined as $\epsilon(X) = \Pr\{X = 0\} - \Pr\{X = 1\}$.

- An n-variable Boolean function $f(\mathbf{x})$ is a mapping from $\mathbf{F}_{2^n}$ to $\mathbf{F}_2$, i.e., $f : \mathbf{F}_{2^n} \to \mathbf{F}_2$.
- The correlation of a Boolean function $f : \mathbf{F}_{2^n} \to \mathbf{F}_2$ to zero is defined as

$$\begin{aligned}\epsilon(f) &= 2^{-n}(|\{\mathbf{x} \in \mathbf{F}_{2^n} : f(\mathbf{x}) = 0\}| - |\{\mathbf{x} \in \mathbf{F}_{2^n} : f(\mathbf{x}) = 1\}|) \\ &= \Pr\{f(X) = 0\} - \Pr\{f(X) = 1\},\end{aligned}$$

  where X is a uniformly distributed random variable in $\mathbf{F}_{2^n}$. Note that "correlation" is often used to evaluate the efficiency of *bitwise* linear approximations in a linear approximation attack, where the data complexity is proportional to $1/\epsilon^2(f)$.
- An (n, m)-function $F(\mathbf{x})$ is a mapping from $\mathbf{F}_{2^n}$ to $\mathbf{F}_{2^m}$, i.e., $F : \mathbf{F}_{2^n} \to \mathbf{F}_{2^m}$ such that $\mathbf{x} \mapsto (f_0, ..., f_{m-1})$, where f_is are n-variable Boolean functions. F is also called an m-dimensional vectorial Boolean function.
- For an (n, m)-function F, the probability distribution D_F of F is $D_F(\mathbf{a}) = \frac{|\{\mathbf{x} \in \mathbf{F}_{2^n} : F(\mathbf{x}) = \mathbf{a}\}|}{2^n}$ for all $\mathbf{a} \in \mathbf{F}_{2^m}$.
- The Squared Euclidean Imbalance (SEI) of a distribution D_F is defined as

$$\Delta(D_F) = 2^m \sum_{\mathbf{a} \in \mathbf{F}_{2^m}} (D_F(\mathbf{a}) - \frac{1}{2^m})^2,$$

  which measures the distance between the target distribution and the uniform distribution. Especially for $m = 1$, $\Delta(D_F)$ is closely related to the correlation of F by $\Delta(D_F) = \epsilon^2(F)$. Note that the "SEI" of a distribution D_F over a general alphabet is used to evaluate the efficiency of *large-unit* linear approximations in a linear approximation attack, where the data complexity is proportional to $1/\Delta(D_F)$.
- The correlation of an (n, m)-function $F : \mathbf{F}_{2^n} \to \mathbf{F}_{2^m}$ with a linear output mask $\mathbf{\Gamma} \in \mathbf{F}_{2^m}$ and a linear input mask $\mathbf{\Lambda} \in \mathbf{F}_{2^n}$ is defined as

$$\epsilon_F(\mathbf{\Gamma}; \mathbf{\Lambda}) = \Pr\{\mathbf{\Gamma} \cdot F(X) = \mathbf{\Lambda} \cdot X\} - \Pr\{\mathbf{\Gamma} \cdot F(X) \neq \mathbf{\Lambda} \cdot X\},$$

  where X is a uniformly distributed random variable in $\mathbf{F}_{2^n}$.

2.2 Description of SNOW-V

SNOW-V is a new proposed member in the SNOW family of stream ciphers. It has kept the general design from SNOW 3G in terms of the LFSR and the FSM, but both components are updated to better align with vectorized implementations, and also the size of the FSM has been increased from 32 bits to 128 bits. The overall schematic of SNOW-V algorithm is shown in Fig. 1. For more details on the design of SNOW-V, we refer to the original design document [8].

The LFSR part consists of two LFSRs, namely LFSR-A and LFSR-B, both of 16 cells of length 16, giving 512 bits in total. Denote the states of the LFSRs at time t as $(a_{t+15}, a_{t+14}, ..., a_t)$ and $(b_{t+15}, b_{t+14}, ..., b_t)$ respectively for LFSR-A and LFSR-B, where a_{t+i} and b_{t+i} represent elements in $\mathbf{F}_{2^{16}}$ defined by different generating polynomials. The elements a_{t+i} of LFSR-A are generated by the polynomial

$$g^A(x) = x^{16} + x^{15} + x^{12} + x^{11} + x^8 + x^3 + x^2 + x + 1 \in \mathbf{F}_2[x]$$

and the elements b_{t+i} of LFSR-B are generated by

$$g^B(x) = x^{16} + x^{15} + x^{14} + x^{11} + x^8 + x^6 + x^5 + x + 1 \in \mathbf{F}_2[x]$$

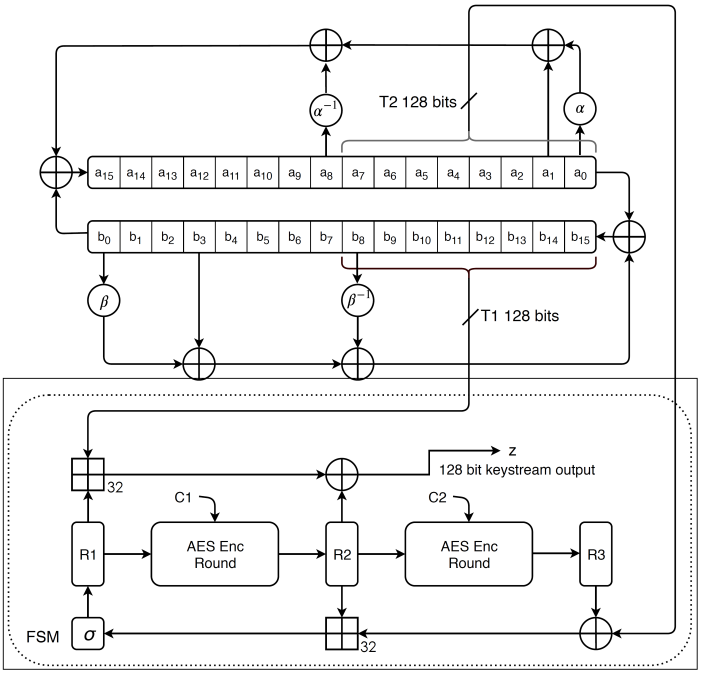

Figure 1: The keystream generation phase of the SNOW-V stream cipher

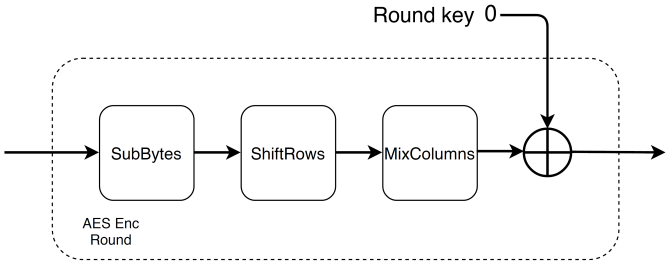

Figure 2: The AES encryption round function AES^R with the round key constant being 0

Let α be a root of $g^A(x)$ and β be a root of $g^B(x)$. The LFSR-A sequence and LFSR-B sequence are given by the expressions $a_{t+16} = b_t \oplus \alpha a_t \oplus a_{t+1} \oplus \alpha^{-1} a_{t+8}$, and $b_{t+16} = a_t \oplus \beta b_t \oplus b_{t+3} \oplus \beta^{-1} b_{t+8}$ respectively, where "$\oplus$" denotes the bitwise XOR of 16-bit blocks.

The FSM of SNOW-V has three 128-bit registers, $R1$, $R2$ and $R3$. Let $T1_t$ be a 128-bit word from the LFSR-B such that

$$T1_t = (b_{8t+15}, b_{8t+14}, ..., b_{8t+8}),$$

and $T2_t$ be a 128-bit word from the LFSR-A such that

$$T2_t = (a_{8t+7}, a_{8t+6}, ..., a_{8t}).$$

Let "$\boxplus_{32}$" denote a parallel application of four additions modulo 2^{32} over each sub-word, and "$\oplus$" denote the bitwise XOR operation of 128-bit blocks. The FSM takes the two blocks $T1_t$ and $T2_t$ as inputs, produces a 128-bit keystream $z_t = (T1_t \boxplus_{32} R1_t) \oplus R2_t$ as

output, and updates the registers $R1$, $R2$ and $R3$ according to

$$\begin{aligned} R1_{t+1} &= \sigma((T2_t \oplus R3_t) \boxplus_{32} R2_t), \\ R2_{t+1} &= AES^R(R1_t), \\ R3_{t+1} &= AES^R(R2_t), \end{aligned}$$

where σ is a byte-wise permutation given by

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

and $AES^R(\cdot)$ denotes a full AES encryption round function with the round key constant being zero, as shown in Fig. 2. Let r be a 128-bit input to $AES^R(\cdot)$, then r is mapped to the state array of the AES round function in the following way:

$$(r_0 \parallel r_1 \parallel r_2 \parallel r_3 \parallel r_4 \parallel r_5 \parallel r_6 \parallel r_7 \parallel r_8 \parallel r_9 \parallel r_{10} \parallel r_{11} \parallel r_{12} \parallel r_{13} \parallel r_{14} \parallel r_{15})$$

$$\Longleftrightarrow \begin{pmatrix} r_0 & r_4 & r_8 & r_{12} \\ r_1 & r_5 & r_9 & r_{13} \\ r_2 & r_6 & r_{10} & r_{14} \\ r_3 & r_7 & r_{11} & r_{15} \end{pmatrix}, r_i \in \mathbf{F}_{2^8}$$

and the output can be written as $\texttt{MixColumns}(\texttt{ShiftRows}(\texttt{SubBytes}(r)))$.

3 Computing the Bitwise Linear Approximations of Certain Types of Functions

In this section, we summarize some algorithms to efficiently compute the linear approximations of certain types of composition functions composed of basic operations like $\boxplus$, $\oplus$, *Permutation*, and *S-box*, by using the slice-like techniques. These functions have been widely used in word-oriented stream ciphers such as SOSEMANUK [2] and SNOW-like ciphers.

3.1 Function Types

(1) **Type-I function**. We define the (n, n)-function $Sbox : \mathbf{F}_{2^n} \to \mathbf{F}_{2^n}$ as the Type-I function, which is constructed by several parallel small-scale s-boxes s_j such that

$$Sbox(\mathbf{x}) = (\mathrm{s}_0(\mathbf{x}_0) \parallel \mathrm{s}_1(\mathbf{x}_1) \parallel ... \parallel \mathrm{s}_{d-1}(\mathbf{x}_{d-1})), \tag{1}$$

where $\mathbf{x} = (\mathbf{x}_0 \parallel \mathbf{x}_1 \parallel ... \parallel \mathbf{x}_{d-1})$ with $\mathbf{x}_j \in \mathbf{F}_{2^m}$ and $d = \frac{n}{m}$, and s_j are all (m, m)-functions, $j = 0, 1, ..., d-1$.

Problem 1. Compute the correlation of the bitwise linear approximation of the Type-I function $Sbox(\cdot)$ with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input mask $\mathbf{\Gamma}^{(1)}$, which is denoted by $\mathrm{Cor}_1(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)})$.

Method and Complexity. The computation of the bitwise linear approximation of the Type-I function is usually carried out according to the preprocessing phase and processing phase. After the preprocessing phase, the processing phase will cost a *linear-time* complexity. The detailed process is described in Appendix A.

(2) **Type-II function**. We define the ρ-input addition modulo 2^n as the Type-II function, i.e., $F : \mathbf{F}_{2^n} \times ... \times \mathbf{F}_{2^n} \to \mathbf{F}_{2^n}$, such that

$$F(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\rho)}) = \mathbf{x}^{(1)} \boxplus_n \cdots \boxplus_n \mathbf{x}^{(\rho)}. \tag{2}$$

Problem 2. Compute the correlation of the bitwise linear approximation of the Type-II function F with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks

$\mathbf{\Gamma}^{(1)}, ..., \mathbf{\Gamma}^{(\rho)}$, which is denoted by $\mathrm{Cor}_2(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, ..., \mathbf{\Gamma}^{(\rho)})$.
Method and Complexity. In [19], the authors have proposed a *linear-time* algorithm to compute the correlation of the bitwise linear approximation of F for any given mask tuple, we describe it in Appendix B.

(3) **Type-III function**. We define the following function $G : \mathbf{F}_{2^n} \times \mathbf{F}_{2^n} \to \mathbf{F}_{2^n}$ as the Type- III function:

$$G(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \mathbf{x}^{(1)} \boxplus_n Sbox(\mathbf{x}^{(2)}), \tag{3}$$

where $Sbox(\cdot)$ is the Type-I function defined above. We emphasize the importance of this type of function, since they are at the core of SNOW ciphers like SNOW 2.0, SNOW 3G and also SNOW-V, as later shown in Section 4.

Problem 3. Compute the correlation of the bitwise linear approximation of the Type-III function G with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks $\mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}$, which is denoted by $\mathrm{Cor}_3(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)})$.
Method and Complexity. In [11], a *linear-time* algorithm is proposed to compute the correlation of the bitwise linear approximation of G under any given mask tuple, and then used to mount attacks on SNOW 2.0 and SNOW 3G. At a very high level, the idea is to divide the n-bit values into d values of m-bit according to the specific structure of the function $Sbox(\cdot)$, and then pre-compute and store some useful matrices, and finally compute the correlation by doing some matrix multiplications using these pre-computed matrices. The details are given in Appendix C.

(4) **Type-IV function**. Let n, m be two positive integers such that m divides n and $d = \frac{n}{m}$. For each variable $\mathbf{x} \in \mathbf{F}_{2^n}$, we can split it into d blocks and each block has m bits, i.e., $\mathbf{x} = (\mathbf{x}_0 \parallel \mathbf{x}_1 \parallel ... \parallel \mathbf{x}_{d-1})$ with $\mathbf{x}_j \in \mathbf{F}_{2^m}$ for $j = 0, 1, ..., d-1$. Let p be a permutation of 0 to $d-1$ and define $p(\mathbf{x}) = (\mathbf{x}_{p(0)} \parallel \mathbf{x}_{p(1)} \parallel ... \parallel \mathbf{x}_{p(d-1)})$ for $\mathbf{x} \in \mathbf{F}_{2^n}$. Based on this, we define the function $H : \mathbf{F}_{2^n} \times \mathbf{F}_{2^n} \times \mathbf{F}_{2^n} \to \mathbf{F}_{2^n}$ as the Type-IV function such that

$$H(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}) = \mathbf{x}^{(1)} \boxplus_n p(\mathbf{x}^{(2)} \boxplus_m \mathbf{x}^{(3)}), \tag{4}$$

which is composed of the addition modulo 2^n ("$\boxplus_n$") , the addition modulo 2^m ("$\boxplus_m$") and the permutation p.

Problem 4. Compute the correlation of the bitwise linear approximation of the Type-IV function H with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks $\mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}$, which is denoted by $\mathrm{Cor}_4(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$.
Method and Complexity. We will show how to compute the correlation of the bitwise linear approximation of the Type-IV function H in Section 3.2.

(5) **Type-V function**. Let l, n, m be three positive integers such that m divides n and n divides l, with $d = \frac{n}{m}$ and $d' = \frac{l}{n}$. For each variable $\mathbf{X} \in \mathbf{F}_{2^l}$, we can split it into d' blocks and each block has n bits, i.e., $\mathbf{X} = (\mathbf{X}_0 \parallel \mathbf{X}_1 \parallel ... \parallel \mathbf{X}_{d'-1})$ with $\mathbf{X}_k \in \mathbf{F}_{2^n}$ for $k = 0, 1, ..., d'-1$, and for each $\mathbf{X}_k \in \mathbf{F}_{2^n}$, it can be split into d blocks and each block has m bits, i.e., $\mathbf{X}_k = (\mathbf{x}_{kd} \parallel \mathbf{x}_{kd+1} \parallel ... \parallel \mathbf{x}_{kd+(d-1)})$ with $\mathbf{x}_{kd+j} \in \mathbf{F}_{2^m}$ for $j = 0, 1, ..., d-1$. Let P be a permutation of 0 to $d'd-1$ such that

$$\begin{aligned} P(\mathbf{X}) = (&\underbrace{\mathbf{x}_{P(0)} \parallel ... \parallel \mathbf{x}_{P(d-1)}} \parallel ... \parallel \underbrace{\mathbf{x}_{P(kd)} \parallel ... \parallel \mathbf{x}_{P(kd+(d-1))}} \parallel ... \\ &\parallel \underbrace{\mathbf{x}_{P((d'-1)d)} \parallel ... \parallel \mathbf{x}_{P(d'd-1)}}) \end{aligned}$$

Based on this, we define the function $\mathcal{H} : \mathbf{F}_{2^l} \times \mathbf{F}_{2^l} \times \mathbf{F}_{2^l} \to \mathbf{F}_{2^l}$ as the Type-V function such that

$$\mathcal{H}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}) = \mathbf{X}^{(1)} \boxplus_n P(\mathbf{X}^{(2)} \boxplus_m \mathbf{X}^{(3)}), \tag{5}$$

which is composed of the addition modulo 2^n ("$\boxplus_n$") , the addition modulo 2^m ("$\boxplus_m$") and the permutation P.

Problem 5. Compute the correlation of the bitwise linear approximation of the Type-V function $\mathcal{H}$ with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks $\mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}$, which is denoted by $\mathrm{Cor}_5(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$.

Method and Complexity. As later shown in Section 5, this type of function plays an important role in analyzing the strength of SNOW-V against linear approximation attacks. We will show how to compute the correlation of the bitwise linear approximation in Section 3.3.

3.2 Computing the Bitwise Linear Approximation of Type-IV Function

Problem 4. Compute the correlation of the bitwise linear approximation of the Type-IV function H with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks $\mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}$, which is denoted by $\mathrm{Cor}_4(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$.

Method and Complexity. Due to the definition of the permutation p, we have

$$p(\mathbf{x}^{(2)} \boxplus_m \mathbf{x}^{(3)}) = ((\mathbf{x}^{(2)}_{p(0)} \boxplus_m \mathbf{x}^{(3)}_{p(0)}) \parallel (\mathbf{x}^{(2)}_{p(1)} \boxplus_m \mathbf{x}^{(3)}_{p(1)}) \parallel ... \parallel (\mathbf{x}^{(2)}_{p(d-1)} \boxplus_m \mathbf{x}^{(3)}_{p(d-1)})),$$

which belongs to the Type-I function with $\mathrm{s}_j(\cdot) = \mathbf{x}^{(2)}_{p(j)} \boxplus_m \mathbf{x}^{(3)}_{p(j)}$ for $j = 0, 1, ..., d-1$. Thus H is actually in the Type-III category, and can be solved using the method in [11] (Theorem 3 in Appendix C). We describe the process as follows.

3.2.1 A Linear Representation of $\mathrm{Cor}_4(\mathbf{\Gamma^{(0)}}; \mathbf{\Gamma^{(1)}}, \mathbf{\Gamma^{(2)}}, \mathbf{\Gamma^{(3)}})$

Similar with the method in Appendix C, we split the masks $\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}$ into d blocks as $\mathbf{\Gamma}^{(i)} = (\mathbf{\Gamma}^{(i)}_0 \parallel \mathbf{\Gamma}^{(i)}_1 \parallel ... \parallel \mathbf{\Gamma}^{(i)}_{d-1})$ with $\mathbf{\Gamma}^{(i)}_j \in \mathbf{F}_{2^m}$. Denote $u_j = \mathbf{x}^{(1)}_j + (\mathbf{x}^{(2)}_{p(j)} \boxplus_m \mathbf{x}^{(3)}_{p(j)})$, $j = 0, 1, ..., d-1$, we define the Boolean functions h_j for $j = 0, 1, ..., d-1$ as follows:

$$h_j(\cdot) = \mathbf{\Gamma}^{(0)}_j \cdot (u_j \boxplus_m \theta_j) \oplus \mathbf{\Gamma}^{(1)}_j \cdot \mathbf{x}^{(1)}_j \oplus \mathbf{\Gamma}^{(2)}_{p(j)} \cdot \mathbf{x}^{(2)}_{p(j)} \oplus \mathbf{\Gamma}^{(3)}_{p(j)} \cdot \mathbf{x}^{(3)}_{p(j)},$$

where $\theta_0 = 0$ and $\theta_{j+1} = \lfloor (u_j + \theta_j)/2^m \rfloor$. Let $\mathcal{V}_j = (\mathbf{\Gamma}^{(0)}_j, \mathbf{\Gamma}^{(1)}_j, \mathbf{\Gamma}^{(2)}_{p(j)}, \mathbf{\Gamma}^{(3)}_{p(j)})$ and $\mathbf{C}_{\mathcal{V}_j}$ be the 2×2 matrix such that

$$\begin{aligned}\mathbf{C}_{\mathcal{V}_j}[\alpha][\theta_j] = 2^{-3m}(&|\{\mathbf{x}^{(1)}_j, \mathbf{x}^{(2)}_{p(j)}, \mathbf{x}^{(3)}_{p(j)} \in \mathbf{F}_{2^m} : h_j(\theta_j, \cdot) = 0,\ \theta_{j+1}(\theta_j, \cdot) = \alpha\}| \\ &- |\{\mathbf{x}^{(1)}_j, \mathbf{x}^{(2)}_{p(j)}, \mathbf{x}^{(3)}_{p(j)} \in \mathbf{F}_{2^m} : h_j(\theta_j, \cdot) = 1,\ \theta_{j+1}(\theta_j, \cdot) = \alpha\}|),\end{aligned}$$

for $\alpha \in \{0, 1\}$ and $\theta_j \in \{0, 1\}$. According to Theorem 3 in Appendix C, we have

$$\mathrm{Cor}_4(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}) = \boldsymbol{l}_2 \mathbf{C}_{\mathcal{V}_{d-1}} ... \mathbf{C}_{\mathcal{V}_1} \mathbf{C}_{\mathcal{V}_0} \boldsymbol{e}_0. \tag{6}$$

That means the correlation of the bitwise linear approximation of H for the given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$ can be accurately computed by doing d multiplications of a 2×2 matrix and a column vector, and one additional addition, which is a *linear-time* procedure.

For a given partial mask tuple $\mathcal{V} = (\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}, \gamma^{(3)})$ with $\gamma^{(i)} \in \mathbf{F}_{2^m}$, we define a general expression for all the h_j as follows:

$$\begin{aligned} h : \{0, 1\} \times (\mathbf{F}_{2^m})^3 &\to \{0, 1\} \times \mathbf{F}_2 \\ (i\theta, \mathbf{x}, \mathbf{y}, \mathbf{z}) &\mapsto (o\theta, r) \end{aligned}$$

such that $o\theta(i\theta,\cdot) = \lfloor(\mathbf{x} + (\mathbf{y} \boxplus_m \mathbf{z}) + i\theta)/2^m\rfloor$ and $r(i\theta,\cdot) = \gamma^{(0)} \cdot (\mathbf{x} \boxplus_m (\mathbf{y} \boxplus_m \mathbf{z}) \boxplus_m i\theta) \oplus \gamma^{(1)} \cdot \mathbf{x} \oplus \gamma^{(2)} \cdot \mathbf{y} \oplus \gamma^{(3)} \cdot \mathbf{z}$, where $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{F}_{2^m}$ and $i\theta \in \{0,1\}$. Thus we have

$$\begin{aligned}\mathbf{C}_{\mathcal{V}}[\alpha][i\theta] = 2^{-3m}(&|\{\mathbf{x},\mathbf{y},\mathbf{z} \in \mathbf{F}_{2^m} : r(i\theta,\cdot) = 0,\ o\theta(i\theta,\cdot) = \alpha\}| \\ &- |\{\mathbf{x},\mathbf{y},\mathbf{z} \in \mathbf{F}_{2^m} : r(i\theta,\cdot) = 1,\ o\theta(i\theta,\cdot) = \alpha\}|).\end{aligned} \tag{7}$$

for $i\theta \in \{0,1\}$ and $\alpha \in \{0,1\}$. Based on this, we say that the row vector $\boldsymbol{l}_2$, the column vector $\boldsymbol{e}_0$ and the matrices $\mathbf{C}_{\mathcal{V}}$ for all 2^{4m} possibilities of $\mathcal{V}$ form a linear representation of the correlation of all the bitwise linear approximations of the Type-IV function H. Now the task remains to compute these matrices.

For any given partial mask tuple $\mathcal{V} = (\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}, \gamma^{(3)})$, the straightforward approach to compute the matrix $\mathbf{C}_{\mathcal{V}}$ by (7) needs a time complexity of $\mathcal{O}(2^{3m+1})$. If our goal is to find all those mask tuples $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$ such that the linear approximations of H would be highly biased, we seem to need to pre-compute the matrices $\mathbf{C}_{\mathcal{V}}$ for all 2^{4m} possibilities of $\mathcal{V}$, and thus the time complexity would be $\mathcal{O}(2^{7m+1})$, which is impractical for the most common value $m = 8$. Fortunately, this can be efficiently solved by adapting the idea of Theorem 2 in Appendix B to our case to compute the matrices $\mathbf{C}_{\mathcal{V}}$ for all the possible values of $\mathcal{V}$.

In the following, we will derive a linear representation for the matrices $\mathbf{C}_{\mathcal{V}}$ by adapting the bit-slicing technique of Theorem 2.

3.2.2 A Linear Representation for the Matrices $\mathbf{C}_{\mathcal{V}}$

Write $\mathbf{x}, \mathbf{y}, \mathbf{z}$ and also $\gamma^{(i)}$ in bits as

$$\begin{aligned}\mathbf{x} &= (x_0 \parallel x_1 \parallel ... \parallel x_{m-1}),\\ \mathbf{y} &= (y_0 \parallel y_1 \parallel ... \parallel y_{m-1}),\\ \mathbf{z} &= (z_0 \parallel z_1 \parallel ... \parallel z_{m-1}),\end{aligned}$$

and $\gamma^{(i)} = (\gamma_0^{(i)} \parallel \gamma_1^{(i)} \parallel ... \parallel \gamma_{m-1}^{(i)})$ for $i = 0,1,2,3$, we have derived the following expressions for $(r, o\theta)$ of the general expression h:

$$\begin{aligned}r(i\theta,\cdot) = {}& \gamma_0^{(0)} \cdot (x_0 \oplus (y_0 \oplus z_0 \oplus 0) \oplus i\theta) \oplus \gamma_0^{(1)} \cdot x_0 \oplus \gamma_0^{(2)} \cdot y_0 \oplus \gamma_0^{(3)} \cdot z_0\\ &\oplus \gamma_1^{(0)} \cdot (x_1 \oplus (y_1 \oplus z_1 \oplus cr_1^0) \oplus cr_1^1) \oplus \gamma_1^{(1)} \cdot x_1 \oplus \gamma_1^{(2)} \cdot y_1 \oplus \gamma_1^{(3)} \cdot z_1\\ &\oplus .....\\ &\oplus \gamma_j^{(0)} \cdot (x_j \oplus (y_j \oplus z_j \oplus cr_j^0) \oplus cr_j^1) \oplus \gamma_j^{(1)} \cdot x_j \oplus \gamma_j^{(2)} \cdot y_j \oplus \gamma_j^{(3)} \cdot z_j\\ &\oplus .....\\ &\oplus \gamma_{m-1}^{(0)} \cdot (x_{m-1} \oplus (y_{m-1} \oplus z_{m-1} \oplus cr_{m-1}^0) \oplus cr_{m-1}^1) \oplus \gamma_{m-1}^{(1)} \cdot x_{m-1}\\ &\oplus \gamma_{m-1}^{(2)} \cdot y_{m-1} \oplus \gamma_{m-1}^{(3)} \cdot z_{m-1},\end{aligned}$$

and $o\theta(i\theta,\cdot) = cr_m^1$, where $cr_j^0, cr_j^1 \in \{0,1\}$ are local carries introduced by two additions modulo 2^m such that

$$\begin{cases} cr_0^0 = 0,\\ cr_{j+1}^0 = \left\lfloor (y_j + z_j + cr_j^0)/2 \right\rfloor, \ j = 0,1,...,m-1 \end{cases}$$

and

$$\begin{cases} cr_0^1 = i\theta,\\ cr_{j+1}^1 = \left\lfloor (x_j + (y_j \oplus z_j \oplus cr_j^0) + cr_j^1)/2 \right\rfloor, \ j = 0,1,...,m-1 \end{cases}$$

We let $r_0, r_1, ..., r_{m-1}$ denote the expressions extracted from r such that

$$r_j(\cdot) = \gamma_j^{(0)} \cdot (x_j \oplus (y_j \oplus z_j \oplus cr_j^0) \oplus cr_j^1) \oplus \gamma_j^{(1)} \cdot x_j \oplus \gamma_j^{(2)} \cdot y_j \oplus \gamma_j^{(3)} \cdot z_j,$$

then we have

$$r(i\theta,\cdot) = r_0(\cdot) \oplus r_1(\cdot) \oplus ... \oplus r_{m-1}(\cdot).$$

Through the above analysis, we describe in the following theorem a linear-time algorithm to compute the matrix $\mathbf{C}_{\mathcal{V}}$ for each given $\mathcal{V}$.

Theorem 1. *Let* $\mathbf{L}$ *be the* 2×4 *constant matrix that* $\mathbf{L} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, *and* $\mathbf{E}$ *be the* 4×2 *constant matrix that* $\mathbf{E} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}^{\mathrm{T}}$, *For any given* $\mathcal{V} = (\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}, \gamma^{(3)})$, *let* $\vec{v}_j = (\gamma_j^{(0)}, \gamma_j^{(1)}, \gamma_j^{(2)}, \gamma_j^{(3)})$, $j = 0, 1, ..., m-1$, *then the* 2×2 *matrix* $\mathbf{C}_{\mathcal{V}}$ *can be computed as:*

$$\mathbf{C}_{\mathcal{V}} = \mathbf{L}\mathbf{c}_{\vec{v}_{m-1}} ... \mathbf{c}_{\vec{v}_1}\mathbf{c}_{\vec{v}_0}\mathbf{E},$$

where $\mathbf{c}_{\vec{v}_j}$ *are* 4×4 *matrices such that*

$$\begin{aligned}
&\mathbf{c}_{\vec{v}_j}[2\alpha^1 + \alpha^0][2cr_j^1 + cr_j^0] \\
&= 2^{-3}(|\{x_j, y_j, z_j \in \mathbf{F}_2 : r_j(cr_j^0, cr_j^1, \cdot) = 0,\ cr_{j+1}^0(cr_j^0, \cdot) = \alpha^0,\ cr_{j+1}^1(cr_j^0, cr_j^1, \cdot) = \alpha^1\}| \\
&\quad - |\{x_j, y_j, z_j \in \mathbf{F}_2 : r_j(cr_j^0, cr_j^1, \cdot) = 1,\ cr_{j+1}^0(cr_j^0, \cdot) = \alpha^0,\ cr_{j+1}^1(cr_j^0, cr_j^1, \cdot) = \alpha^1\}|),
\end{aligned}$$

for $\alpha^0, \alpha^1 \in \{0,1\}$ *and* $cr_j^0, cr_j^1 \in \{0,1\}$, *which are pre-computed by Algorithm 2. We say that the constant matrices* $\mathbf{L}$ *and* $\mathbf{E}$, *and the* 4×4 *matrices* $\mathbf{c}_{\vec{v}}$ *for all* $\vec{v} \in \mathbf{F}_2^4$ *form a linear representation for all the* 2^{4m} *matrices* $\mathbf{C}_{\mathcal{V}}$.

Algorithm 2 Construction of the matrices $\mathbf{c}_{\vec{v}}$ for all $\vec{v}$

Prepare two 4×4 matrices $\mathbf{N}_0$ and $\mathbf{N}_1$;
for $\vec{v} = (v^{(0)}, v^{(1)}, v^{(2)}, v^{(3)}) \in \mathbf{F}_2^4$ **do**
  Create a matrix $\mathbf{c}_{\vec{v}}$ of size 4×4;
  Initialize $\mathbf{N}_0$ and $\mathbf{N}_1$ with zeros;
  for $ic^0 \in \{0,1\}$, $ic^1 \in \{0,1\}$ and $x, y, z \in \mathbf{F}_2$ **do**
    compute $r = v^{(0)} \cdot [x \oplus (y \oplus z \oplus ic^0) \oplus ic^1] \oplus v^{(1)} \cdot x \oplus v^{(2)} \cdot y \oplus v^{(3)} \cdot z$;
    compute $oc^0 = \lfloor (y + z + ic^0)/2 \rfloor$;
    compute $oc^1 = \lfloor (x + (y \oplus z \oplus ic^0) + ic^1)/2 \rfloor$;
    $\mathbf{N}_r[2\cdot oc^1 + oc^0][2\cdot ic^1 + ic^0] := \mathbf{N}_r[2\cdot oc^1 + oc^0][2\cdot ic^1 + ic^0] + 1$;
  end for
  for $ic^0, ic^1 \in \{0,1\}$ and $oc^0, oc^1 \in \{0,1\}$ **do**
    $\mathbf{c}_{\vec{v}}[2\cdot oc^1 + oc^0][2\cdot ic^1 + ic^0]$
    $:= (\mathbf{N}_0[2\cdot oc^1 + oc^0][2\cdot ic^1 + ic^0] - \mathbf{N}_1[2\cdot oc^1 + oc^0][2\cdot ic^1 + ic^0])/2^3$;
  end for
end for

Output: all the matrices $\mathbf{c}_{\vec{v}}$ for all 16 possible values of $\vec{v}$.

Remark. From Algorithm 2, all the matrices $\mathbf{c}_{\vec{v}}$ can be constructed with a total time complexity of about $2^4 \times 2^5 = 2^9$ and a total memory complexity of $2^4 \times (4\times 4) = 2^8$. For a fixed $\mathcal{V}$, the matrix $\mathbf{C}_{\mathcal{V}}$ can be computed according to Theorem 1 by doing m matrix multiplications of small size, by utilizing the matrices $\mathbf{c}_{\vec{v}_j}$ and the constant matrices $\mathbf{L}$ and $\mathbf{E}$, which needs only a linear-time complexity of $\mathcal{O}(m)$, whereas the straightforward method by Eq. (7) would require a complexity of $\mathcal{O}(2^{3m+1})$.

Now we summarize the general process for computing the correlation of the bitwise linear approximation of H for any given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$ in the following two phases:

- In the preprocessing phase, we pre-compute the matrices $\mathbf{C}_{\mathcal{V}}$ for all (at most) 2^{4m} possible values of $\mathcal{V}$ according to Theorem 1, which requires a total time complexity of $\mathcal{O}(m2^{4m})$ and a total memory complexity of $\mathcal{O}(2^{4m})$ for all $\mathcal{V}$.

- In the processing phase, we compute the correlation of the linear approximation of H for each given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$ according to (6), i.e., $\mathrm{Cor}_4(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}) = \boldsymbol{l}_2 \mathbf{C}_{\mathcal{V}_{d-1}} ... \mathbf{C}_{\mathcal{V}_1} \mathbf{C}_{\mathcal{V}_0} \boldsymbol{e}_0$, by doing d multiplications of a 2×2 matrix and a column vector, and one additional addition, which is a *linear-time* procedure, and thus allows for a wide range of search of highly biased linear approximations.

3.3 Computing the Bitwise Linear Approximation of Type-V Function

Problem 5. Compute the correlation of the bitwise linear approximation of the Type-V function $\mathcal{H}$ with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks $\mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}$, which is denoted by $\mathrm{Cor}_5(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$.

Method and Complexity. Note that $\mathcal{H}$ looks quite similar to the Type-IV function H. We will present a similar method to compute the bitwise linear approximation of $\mathcal{H}$ by utilizing the pre-computed matrices $\mathbf{C}_{\mathcal{V}}$ in Section 3.2. Considering the structure of $\mathcal{H}$, we represent the masks $\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}$ as

$$\mathbf{\Gamma}^{(i)} = (\underbrace{\mathbf{\Gamma}^{(i)}_0 \parallel ... \parallel \mathbf{\Gamma}^{(i)}_{d-1}} \parallel ... \parallel \underbrace{\mathbf{\Gamma}^{(i)}_{kd} \parallel ... \parallel \mathbf{\Gamma}^{(i)}_{kd+(d-1)}} \parallel ... \parallel \underbrace{\mathbf{\Gamma}^{(i)}_{(d'-1)d} \parallel ... \parallel \mathbf{\Gamma}^{(i)}_{d'd-1}}),$$

where $\mathbf{\Gamma}^{(i)}_{kd+j} \in \mathbf{F}_{2^m}$ for $i = 0, 1, 2, 3$, $j = 0, ..., d-1$ and $k = 0, ..., d'-1$. Let $\mathcal{V}_{kd+j} = (\mathbf{\Gamma}^{(0)}_{kd+j}, \mathbf{\Gamma}^{(1)}_{kd+j}, \mathbf{\Gamma}^{(2)}_{P(kd+j)}, \mathbf{\Gamma}^{(3)}_{P(kd+j)}) \in (\mathbf{F}_{2^m})^4$ be vectors defined according to $\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}$, and $\mathbf{C}_{\mathcal{V}_{kd+j}}$ be the matrices pre-computed according to Theorem 1. By adapting Equation (6) for computing the bitwise linear approximation of $\mathcal{H}$, we deduce that

$$\mathrm{Cor}_5(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)}) = \prod_{k=0}^{d'-1} (\boldsymbol{l}_2 \mathbf{C}_{\mathcal{V}_{kd+(d-1)}} ... \mathbf{C}_{\mathcal{V}_{kd+1}} \mathbf{C}_{\mathcal{V}_{kd}} \boldsymbol{e}_0), \tag{8}$$

thus the correlation of the bitwise linear approximation of $\mathcal{H}$ for any given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}, \mathbf{\Gamma}^{(3)})$ can be obtained according to Equation (8) by doing $d'd$ matrix multiplications of small size, which is a *linear-time* procedure.

4 Analysis of SNOW-V When Assuming σ to Be Identity

We consider a general approach to analyze SNOW-V against linear approximation attacks. That is, we try to approximate the FSM part through linear masking and then to cancel out the contributions of the registers $R1$, $R2$ and $R3$ by combining expressions for several keystream outputs. In this section, we assume σ to be identity (denoted by σ_0), i.e., there is no byte-wise permutation in the FSM part, as depicted in Fig. 3, which is also the original version of SNOW-V appearing on the IACR ePrint on November 29, 2018. We denote it SNOW-V$_{\sigma_0}$ to make a distinction. We will first study the bitwise linear approximations for the FSM of SNOW-V$_{\sigma_0}$ by using the linear-time algorithms in Section 3, and then present a bitwise fast correlation attack accordingly.

4.1 Bitwise Linear Approximation of the FSM of SNOW-V$_{\sigma_0}$

To build the bitwise linear approximation of the FSM of SNOW-V$_{\sigma_0}$, we consider to apply the 128-bit linear masks $\mathbf{\Phi}$, $\mathbf{\Gamma}$ and $\mathbf{\Lambda}$ to z_{t-1}, z_t and z_{t+1} respectively, i.e.,

$$\begin{aligned}
\mathbf{\Phi} \cdot z_{t-1} &= \mathbf{\Phi} \cdot (T1_{t-1} \boxplus_{32} R1_{t-1}) \oplus \mathbf{\Phi} \cdot R2_{t-1}, \\
\mathbf{\Gamma} \cdot z_t &= \mathbf{\Gamma} \cdot (T1_t \boxplus_{32} R1_t) \oplus \mathbf{\Gamma} \cdot R2_t, \\
\mathbf{\Lambda} \cdot z_{t+1} &= \mathbf{\Lambda} \cdot (T1_{t+1} \boxplus_{32} R1_{t+1}) \oplus \mathbf{\Lambda} \cdot R2_{t+1}.
\end{aligned}$$

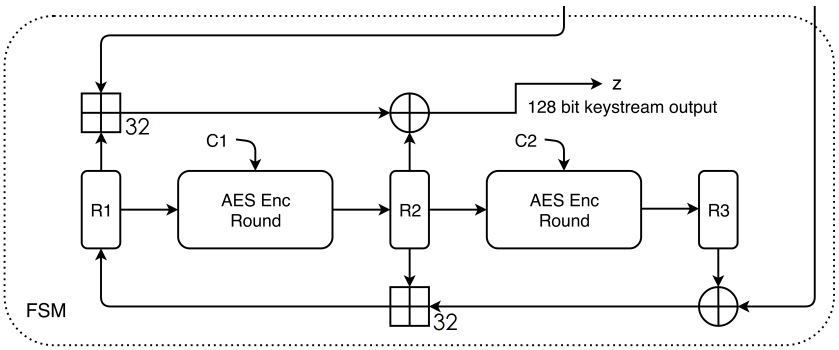

Figure 3: The FSM part of SNOW-V$_{\sigma_0}$

Let $u_t = R1_{t-1}$, $v_t = R2_{t-1}$ and $w_t = R1_t$. According to the update expressions for the registers of the FSM, when σ is just the identity, the first register $R1$ is updated according to $R1_{t+1} = (T2_t \oplus R3_t) \boxplus_{32} R2_t$. We then have $R1_{t+1} = (T2_t \oplus AES^R(v_t)) \boxplus_{32} AES^R(u_t)$, $R2_t = AES^R(u_t)$, and $R2_{t+1} = AES^R(w_t)$, and thus

$$\begin{aligned}
\boldsymbol{\Phi} \cdot z_{t-1} &= \boldsymbol{\Phi} \cdot (T1_{t-1} \boxplus_{32} u_t) \oplus \boldsymbol{\Phi} \cdot v_t, \\
\boldsymbol{\Gamma} \cdot z_t &= \boldsymbol{\Gamma} \cdot (T1_t \boxplus_{32} w_t) \oplus \boldsymbol{\Gamma} \cdot AES^R(u_t), \\
\boldsymbol{\Lambda} \cdot z_{t+1} &= \boldsymbol{\Lambda} \cdot (T1_{t+1} \boxplus_{32} (T2_t \oplus AES^R(v_t)) \boxplus_{32} AES^R(u_t)) \oplus \boldsymbol{\Lambda} \cdot AES^R(w_t).
\end{aligned}$$

Regarding to the internal states and keystream outputs, we consider the following four associated bitwise linear approximations by introducing a 128-bit intermediate linear mask $\boldsymbol{\Theta}$, and write them as follows:

$(\boldsymbol{a})$ $\boldsymbol{\Phi} \cdot (T1_{t-1} \boxplus_{32} u_t) = \boldsymbol{\Phi} \cdot T1_{t-1} \oplus \boldsymbol{\Theta} \cdot AES^R(u_t) \oplus n_a^{(t)}$,

$(\boldsymbol{b})$ $\boldsymbol{\Gamma} \cdot (T1_t \boxplus_{32} w_t) = \boldsymbol{\Gamma} \cdot T1_t \oplus \boldsymbol{\Lambda} \cdot AES^R(w_t) \oplus n_b^{(t)}$,

$(\boldsymbol{c})$ $\boldsymbol{\Lambda} \cdot (T1_{t+1} \boxplus_{32} (T2_t \oplus AES^R(v_t)) \boxplus_{32} AES^R(u_t))$
$= \boldsymbol{\Lambda} \cdot T1_{t+1} \oplus \boldsymbol{\Lambda} \cdot (T2_t \oplus AES^R(v_t)) \oplus (\boldsymbol{\Theta} \oplus \boldsymbol{\Gamma}) \cdot AES^R(u_t) \oplus n_c^{(t)}$,

$(\boldsymbol{d})$ $\boldsymbol{\Lambda} \cdot AES^R(v_t) = \boldsymbol{\Phi} \cdot v_t \oplus n_d^{(t)}$.

where $n_a^{(t)}$ $n_b^{(t)}, n_c^{(t)}, n_d^{(t)}$ are noises introduced by these linear approximations. Let $n^{(t)} = n_a^{(t)} \oplus n_b^{(t)} \oplus n_c^{(t)} \oplus n_d^{(t)}$. With the above relations, the bitwise linear approximations of the FSM of SNOW-V$_{\sigma_0}$ have the following form:

$$\boldsymbol{\Phi} \cdot z_{t-1} \oplus \boldsymbol{\Gamma} \cdot z_t \oplus \boldsymbol{\Lambda} \cdot z_{t+1} = \boldsymbol{\Phi} \cdot T1_{t-1} \oplus \boldsymbol{\Gamma} \cdot T1_t \oplus \boldsymbol{\Lambda} \cdot (T1_{t+1} \oplus T2_t) \oplus n^{(t)}. \quad (9)$$

Basically, we first want to find mask tuples $(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})$ for (9) such that $n^{(t)}$ would be highly biased, and then employ them in a bitwise fast correlation attack. For the four linear approximation relations $(\boldsymbol{a})$, $(\boldsymbol{b})$, $(\boldsymbol{c})$ and $(\boldsymbol{d})$, we have the following illustrations.

1. For the linear approximation relation $(\boldsymbol{a})$, we write

$$AES^R(u_t) = \texttt{MixColumns}(\texttt{ShiftRows}(\texttt{SubBytes}(u_t))).$$

Denote $x_t = \texttt{SubBytes}(u_t)$ as the output of 16 parallel AES S-boxes S_R, then $AES^R(u_t) = \texttt{MixColumns}(\texttt{ShiftRows}(x_t))$. Given the mask $\boldsymbol{\Theta}$, we let $\boldsymbol{\Theta}'$ be the mask such that $\boldsymbol{\Theta}' \cdot x_t = \boldsymbol{\Theta} \cdot \texttt{MixColumns}(\texttt{ShiftRows}(x_t))$. Accordingly we can rewrite the noise $n_a^{(t)}$ as

$$n_a^{(t)} = \boldsymbol{\Phi} \cdot (T1_{t-1} \boxplus_{32} \texttt{SubBytes}^{-1}(x_t)) \oplus \boldsymbol{\Phi} \cdot T1_{t-1} \oplus \boldsymbol{\Theta}' \cdot x_t,$$

where $\texttt{SubBytes}^{-1}(\cdot)$ denotes the inverse of $\texttt{SubBytes}(\cdot)$, i.e., 16 parallel operations of S_R^{-1}. We refer to Appendix D for the computation of $\boldsymbol{\Theta}'$ from $\boldsymbol{\Theta}$.

2. Similarly, for the linear approximation relation ($\boldsymbol{b}$), we let $y_t = \texttt{SubBytes}(w_t)$, and $\boldsymbol{\Lambda}'$ be the mask such that $\boldsymbol{\Lambda}' \cdot y_t = \boldsymbol{\Lambda} \cdot \texttt{MixColumns}(\texttt{ShiftRows}(y_t))$, then the noise $n_b^{(t)}$ can be expressed as

$$n_b^{(t)} = \boldsymbol{\Gamma} \cdot (T1_t \boxplus_{32} \texttt{SubBytes}^{-1}(y_t)) \oplus \boldsymbol{\Gamma} \cdot T1_t \oplus \boldsymbol{\Lambda}' \cdot y_t.$$

3. For the linear approximation relation ($\boldsymbol{c}$), let $\xi_t = T2_t \oplus AES^R(v_t)$ and $\eta_t = AES^R(u_t)$, then the noise $n_c^{(t)}$ can be expressed as

$$n_c^{(t)} = \boldsymbol{\Lambda} \cdot (T1_{t+1} \boxplus_{32} \xi_t \boxplus_{32} \eta_t) \oplus \boldsymbol{\Lambda} \cdot T1_{t+1} \oplus \boldsymbol{\Lambda} \cdot \xi_t \oplus (\boldsymbol{\Theta} \oplus \boldsymbol{\Gamma}) \cdot \eta_t.$$

4. For the linear approximation relation ($\boldsymbol{d}$), we note $\boldsymbol{\Lambda} \cdot AES^R(v_t) = \boldsymbol{\Lambda}' \cdot \texttt{SubBytes}(v_t)$, thus

$$n_d^{(t)} = \boldsymbol{\Lambda}' \cdot \texttt{SubBytes}(v_t) \oplus \boldsymbol{\Phi} \cdot v_t.$$

To sum up, the four linear approximations can be rewritten as follows:

$$\begin{aligned}
n_a^{(t)} &= \boldsymbol{\Phi} \cdot (T1_{t-1} \boxplus_{32} \texttt{SubBytes}^{-1}(x_t)) \oplus \boldsymbol{\Phi} \cdot T1_{t-1} \oplus \boldsymbol{\Theta}' \cdot x_t, \\
n_b^{(t)} &= \boldsymbol{\Gamma} \cdot (T1_t \boxplus_{32} \texttt{SubBytes}^{-1}(y_t)) \oplus \boldsymbol{\Gamma} \cdot T1_t \oplus \boldsymbol{\Lambda}' \cdot y_t, \\
n_c^{(t)} &= \boldsymbol{\Lambda} \cdot (T1_{t+1} \boxplus_{32} \xi_t \boxplus_{32} \eta_t) \oplus \boldsymbol{\Lambda} \cdot T1_{t+1} \oplus \boldsymbol{\Lambda} \cdot \xi_t \oplus (\boldsymbol{\Theta} \oplus \boldsymbol{\Gamma}) \cdot \eta_t, \\
n_d^{(t)} &= \boldsymbol{\Lambda}' \cdot \texttt{SubBytes}(v_t) \oplus \boldsymbol{\Phi} \cdot v_t.
\end{aligned}$$

Since the distributions of the noises $n_a^{(t)}$, $n_b^{(t)}$, $n_c^{(t)}$, $n_d^{(t)}$ are independent of the time instance t, we will simplify them by writing n_a, n_b, n_c, n_d respectively. Let $\epsilon_{\mathrm{FSM}}(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})$ denote the correlation of the linear approximation relation (9) corresponding to the linear mask tuple $(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})$. By applying the results about correlations over composition functions in [18], we have

$$\epsilon_{\mathrm{FSM}}(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda}) = \epsilon(n_b)\epsilon(n_d) \sum\nolimits_{\boldsymbol{\Theta}} \epsilon(n_a)\epsilon(n_c).$$

In the next part, we will show how to compute the correlations of the above noise terms n_a, n_b, n_c and n_d respectively, and finally obtain $\epsilon_{\mathrm{FSM}}(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})$.

4.2 Computation of $\epsilon_{\mathrm{FSM}}(\Phi, \Gamma, \Lambda)$

4.2.1 Computation of the Correlations of n_a and n_b

Note that n_a and n_b have the same form but different 128-bit linear mask tuples, which is $(\boldsymbol{\Phi}; \boldsymbol{\Phi}, \boldsymbol{\Theta}')$ for n_a and $(\boldsymbol{\Gamma}; \boldsymbol{\Gamma}, \boldsymbol{\Lambda}')$ for n_b. Let $\mathcal{G} : \mathbf{F}_{2^{128}} \times \mathbf{F}_{2^{128}} \to \mathbf{F}_{2^{128}}$ be a vectorial Boolean function such that

$$\mathcal{G}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = \mathbf{X}^{(1)} \boxplus_{32} \texttt{SubBytes}^{-1}(\mathbf{X}^{(2)}),$$

where $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ are both 128-bit (4-word) random variables. Let $\mathbf{X}^{(1)} = (\mathbf{x}_0^{(1)} \parallel \mathbf{x}_1^{(1)} \parallel \mathbf{x}_2^{(1)} \parallel \mathbf{x}_3^{(1)})$ and $\mathbf{X}^{(2)} = (\mathbf{x}_0^{(2)} \parallel \mathbf{x}_1^{(2)} \parallel \mathbf{x}_2^{(2)} \parallel \mathbf{x}_3^{(2)})$, where $\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)} \in \mathbf{F}_{2^{32}}$ for all $k = 0, 1, 2, 3$. We define another function G as follows:

$$\begin{aligned}
G : \mathbf{F}_{2^{32}} \times \mathbf{F}_{2^{32}} &\to \mathbf{F}_{2^{32}} \\
(\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}) &\mapsto \mathbf{x}_k^{(1)} \boxplus_{32} \mathrm{Sbox}^{-1}(\mathbf{x}_k^{(2)}),
\end{aligned}$$

where $\mathrm{Sbox}^{-1}(\cdot)$ represents the output of four parallel operations of S_R^{-1}. Then G belongs to the **Type-III** function defined in Section 3.1 with the parameters $n = 32$, $m = 8$ and $d = 4$. Note that the operation "$\boxplus_{32}$" in $\mathcal{G}(\cdot)$ is a parallel application of four additions modulo 2^{32}

over each 32-bit sub-word. It is easy to verify that $\mathcal{G}(\cdot)$ can be expressed as four functions G in parallel. Based on this, we let $\mathbf{\Phi} = (\mathbf{\Phi}_0 \parallel \mathbf{\Phi}_1 \parallel \mathbf{\Phi}_2 \parallel \mathbf{\Phi}_3)$, $\mathbf{\Theta}' = (\mathbf{\Theta}'_0 \parallel \mathbf{\Theta}'_1 \parallel \mathbf{\Theta}'_2 \parallel \mathbf{\Theta}'_3)$, $\mathbf{\Gamma} = (\mathbf{\Gamma}_0 \parallel \mathbf{\Gamma}_1 \parallel \mathbf{\Gamma}_2 \parallel \mathbf{\Gamma}_3)$, $\mathbf{\Lambda}' = (\mathbf{\Lambda}'_0 \parallel \mathbf{\Lambda}'_1 \parallel \mathbf{\Lambda}'_2 \parallel \mathbf{\Lambda}'_3)$, where $\mathbf{\Phi}_k, \mathbf{\Theta}'_k, \mathbf{\Gamma}_k, \mathbf{\Lambda}'_k \in \mathbf{F}_{2^{32}}$ for $k = 0, 1, 2, 3$. Then the correlation of n_a with the mask tuple $(\mathbf{\Phi}; \mathbf{\Phi}, \mathbf{\Theta}')$ and the correlation of n_b with the mask tuple $(\mathbf{\Gamma}; \mathbf{\Gamma}, \mathbf{\Lambda}')$ can be computed according to the Piling-up lemma [17] as follows:

$$\epsilon(n_a) = \prod_{k=0}^{3} \mathrm{Cor}_3(\mathbf{\Phi}_k; \mathbf{\Phi}_k, \mathbf{\Theta}'_k) \triangleq \prod_{k=0}^{3} \epsilon_a^{(k)}, \quad \epsilon(n_b) = \prod_{k=0}^{3} \mathrm{Cor}_3(\mathbf{\Gamma}_k; \mathbf{\Gamma}_k, \mathbf{\Lambda}'_k) \triangleq \prod_{k=0}^{3} \epsilon_b^{(k)}.$$

Complexity. For any 32-bit masks $\mathbf{A}, \mathbf{B}$, we write $\mathbf{A} = (\mathbf{a}_0 \parallel \mathbf{a}_1 \parallel \mathbf{a}_2 \parallel \mathbf{a}_3)$ and $\mathbf{B} = (\mathbf{b}_0 \parallel \mathbf{b}_1 \parallel \mathbf{b}_2 \parallel \mathbf{b}_3)$, where $\mathbf{a}_j, \mathbf{b}_j \in \mathbf{F}_{2^8}$ for $j = 0, 1, 2, 3$, then $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B})$ can be efficiently computed by Theorem 3 in Appendix C as

$$\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B}) = l_2 \mathbf{M}_{(\mathbf{a}_3, \mathbf{a}_3, \mathbf{b}_3)} \mathbf{M}_{(\mathbf{a}_2, \mathbf{a}_2, \mathbf{b}_2)} \mathbf{M}_{(\mathbf{a}_1, \mathbf{a}_1, \mathbf{b}_1)} \mathbf{M}_{(\mathbf{a}_0, \mathbf{a}_0, \mathbf{b}_0)} \boldsymbol{e}_0,$$

where the matrices[1] $\mathbf{M}_{(\mathbf{a}_j, \mathbf{a}_j, \mathbf{b}_j)}$ are pre-computed by Algorithm 1 in Appendix C by setting all the parallel functions s_j to be S_R^{-1}. Generally, to compute $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B})$ for any $\mathbf{A}, \mathbf{B} \in \mathbf{F}_{2^{32}}$, we first pre-compute $2^8 \times 2^8 = 2^{16}$ matrices $\mathbf{M}_{(\alpha, \alpha, \beta)}$ by trying all the possibilities of $\alpha, \beta \in \mathbf{F}_{2^8}$, which requires a time complexity of $2^{16} \times (2^{16} \times 2) = 2^{33}$ and a memory complexity of $2^{16} \times (2 \times 2) = 2^{18}$ according to Algorithm 1. Using these pre-computed matrices, $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B})$ can be obtained by doing 4 matrix multiplications of small size, and thus the value of $\epsilon(n_a)$ with any given $(\mathbf{\Phi}; \mathbf{\Phi}, \mathbf{\Theta}')$(*Resp.* $\epsilon(n_b)$ with any given $(\mathbf{\Gamma}; \mathbf{\Gamma}, \mathbf{\Lambda}')$) can be easily derived.

For the value of $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B})$ under any given masks $\mathbf{A}, \mathbf{B}$, we have the following conclusion which will help in finding good linear approximations for the FSM of SNOW-V$_{\sigma_0}$. The proof is given in Appendix E.

Corollary 1. *For any 32-bit masks* $\mathbf{A} = (\mathbf{a}_0 \parallel \mathbf{a}_1 \parallel \mathbf{a}_2 \parallel \mathbf{a}_3)$ *and* $\mathbf{B} = (\mathbf{b}_0 \parallel \mathbf{b}_1 \parallel \mathbf{b}_2 \parallel \mathbf{b}_3)$, *suppose* $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B}) \neq 0$. *Then we have*

(1) $\mathbf{a}_3 =$`0x00` *if and only if* $\mathbf{b}_3 =$`0x00`*;*
(2) $\mathbf{a}_3 = \mathbf{a}_2 =$`0x00` *if and only if* $\mathbf{b}_3 = \mathbf{b}_2 =$`0x00`*;*
(3) $\mathbf{a}_3 = \mathbf{a}_2 = \mathbf{a}_1 =$`0x00` *if and only if* $\mathbf{b}_3 = \mathbf{b}_2 = \mathbf{b}_1 =$`0x00`*;*
(4) $\mathbf{a}_3 = \mathbf{a}_2 = \mathbf{a}_1 = \mathbf{a}_0 =$`0x00` *if and only if* $\mathbf{b}_3 = \mathbf{b}_2 = \mathbf{b}_1 = \mathbf{b}_0 =$`0x00`.
(5) If $\mathbf{A} = \mathbf{B} =$`0x00000000`, *then* $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B}) = 1$.

4.2.2 Computation of the Correlation of n_c

Let $\mathcal{F} : \mathbf{F}_{2^{128}} \times \mathbf{F}_{2^{128}} \times \mathbf{F}_{2^{128}} \to \mathbf{F}_{2^{128}}$ be a vectorial Boolean function such that

$$\mathcal{F}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}) = \mathbf{X}^{(1)} \boxplus_{32} \mathbf{X}^{(2)} \boxplus_{32} \mathbf{X}^{(3)},$$

where $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, $\mathbf{X}^{(3)}$ are 128-bit (4-word) random variables, then $\mathcal{F}$ can be viewed as a parallel application of four **Type-II** functions defined in Section 3.1 with the parameters $n = 32$ and $\rho = 3$.

For n_c, we denote the involved masks $\mathbf{\Lambda}$, $\mathbf{\Theta}$, $\mathbf{\Gamma}$ as $\mathbf{\Lambda} = (\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3)$, $\mathbf{\Gamma} = (\mathbf{\Gamma}_0 \parallel \mathbf{\Gamma}_1 \parallel \mathbf{\Gamma}_2 \parallel \mathbf{\Gamma}_3)$, $\mathbf{\Theta} = (\mathbf{\Theta}_0 \parallel \mathbf{\Theta}_1 \parallel \mathbf{\Theta}_2 \parallel \mathbf{\Theta}_3)$, where $\mathbf{\Lambda}_k, \mathbf{\Gamma}_k, \mathbf{\Theta}_k \in \mathbf{F}_{2^{32}}$. Then the correlation of n_c with the mask tuple $(\mathbf{\Lambda}; \mathbf{\Lambda}, \mathbf{\Lambda}, \mathbf{\Theta} \oplus \mathbf{\Gamma})$ can be computed as follows:

$$\epsilon(n_c) = \prod_{k=0}^{3} \mathrm{Cor}_2(\mathbf{\Lambda}_k; \mathbf{\Lambda}_k, \mathbf{\Lambda}_k, \mathbf{\Theta}_k \oplus \mathbf{\Gamma}_k) \triangleq \prod_{k=0}^{3} \epsilon_c^{(k)}.$$

[1] Note that we have simplified the representations of the matrices $\mathbf{M}_{(j,(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)}))}$ by $\mathbf{M}_{(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)})}$, since all the parallel functions $\mathrm{s}_j(\cdot)$ are the same in G which is S_R^{-1}.

Complexity. For any 32-bit mask $\mathbf{A}, \mathbf{B}$, we use the approach in Appendix B to compute $\mathrm{Cor}_2(\mathbf{A}; \mathbf{A}, \mathbf{A}, \mathbf{B})$. For this computation, we only need to pre-compute four 3×3 matrices $\mathbf{D}_{(\alpha,\alpha;\alpha;\beta)}$ for $\alpha, \beta \in \mathbf{F}_2$, corresponding to a time complexity of $4 \times (2^3 \times 3) = 2^{6.58}$ and a memory complexity of $4 \times (3 \times 3) = 2^{5.17}$. Let $\mathbf{A} = (a_0 \parallel a_1 \parallel ... \parallel a_{31})$ and $\mathbf{B} = (b_0 \parallel b_1 \parallel ... \parallel b_{31})$, where $a_j, b_j \in \mathbf{F}_2$ for $j = 0, 1, ..., 31$, then $\mathrm{Cor}_2(\mathbf{A}; \mathbf{A}, \mathbf{A}, \mathbf{B})$ can be efficiently computed according to Theorem 2 as

$$\mathrm{Cor}_2(\mathbf{A}; \mathbf{A}, \mathbf{A}, \mathbf{B}) = \boldsymbol{l}_3 \mathbf{D}_{(a_{31},a_{31},a_{31},b_{31})} ... \mathbf{D}_{(a_1,a_1,a_1,b_1)} \mathbf{D}_{(a_0,a_0,a_0,b_0)} \boldsymbol{e}_0,$$

which can be obtained by doing 32 matrix multiplications of small size, and thus the value of $\epsilon(n_c)$ with any given mask tuple $(\boldsymbol{\Lambda}; \boldsymbol{\Lambda}, \boldsymbol{\Lambda}, \boldsymbol{\Theta} \oplus \boldsymbol{\Gamma})$ can be easily derived.

4.2.3 Computation of the Correlation of n_d

Note that $n_d^{(t)} = \boldsymbol{\Lambda}' \cdot \mathtt{SubBytes}(v_t) \oplus \boldsymbol{\Phi} \cdot v_t$, where $\mathtt{SubBytes}(\cdot)$ is an application of 16 AES S-boxes S_R, and can also be represented as four parallel applications of the function $\mathrm{Sbox}(\cdot)$, which is the **Type-I** function with the parameters $n = 32$ and $m = 8$. Then the correlation of n_d with the mask tuple $(\boldsymbol{\Lambda}'; \boldsymbol{\Phi})$ can be computed as

$$\epsilon(n_d) = \prod_{k=0}^{3} \mathrm{Cor}_1(\boldsymbol{\Lambda}'_k; \boldsymbol{\Phi}_k) \triangleq \prod_{k=0}^{3} \epsilon_d^{(k)}.$$

Complexity. We use the approach in Appendix A to compute $\mathrm{Cor}_1(\mathbf{A}; \mathbf{B})$ for any 32-bit masks $\mathbf{A}, \mathbf{B}$. First, we pre-compute a linear approximation table (LAT) to store all the linear approximations of S_R by trying all the possibilities of $\mathbf{a}, \mathbf{b}$ values, i.e., all the values of $\epsilon_{S_R}(\mathbf{a}; \mathbf{b})$ for all $\mathbf{a}, \mathbf{b} \in \mathbf{F}_{2^8}$ are stored in the row of LAT indexed by $(\mathbf{a}, \mathbf{b})$. The pre-computation takes time 2^{24}. Let $\mathbf{A} = (\mathbf{a}_0 \parallel \mathbf{a}_1 \parallel \mathbf{a}_2 \parallel \mathbf{a}_3)$ and $\mathbf{B} = (\mathbf{b}_0 \parallel \mathbf{b}_1 \parallel \mathbf{b}_2 \parallel \mathbf{b}_3)$, where $\mathbf{a}_j, \mathbf{b}_j \in \mathbf{F}_{2^8}$ for $j = 0, 1, 2, 3$, we have

$$\mathrm{Cor}_1(\mathbf{A}; \mathbf{B}) = \epsilon_{S_R}(\mathbf{a}_3; \mathbf{b}_3) \epsilon_{S_R}(\mathbf{a}_2; \mathbf{b}_2) \epsilon_{S_R}(\mathbf{a}_1; \mathbf{b}_1) \epsilon_{S_R}(\mathbf{a}_0; \mathbf{b}_0),$$

which can be derived by table lookups four times, and thus the value of $\epsilon(n_d)$ with any given mask tuple $(\boldsymbol{\Lambda}'; \boldsymbol{\Phi})$ can be directly derived.

4.2.4 Computation of $\epsilon_{\mathrm{FSM}}(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})$

From the above discussion, we have $\epsilon(n_a) = \prod_{k=0}^{3} \epsilon_a^{(k)}$, $\epsilon(n_b) = \prod_{k=0}^{3} \epsilon_b^{(k)}$, $\epsilon(n_c) = \prod_{k=0}^{3} \epsilon_c^{(k)}$, and $\epsilon(n_d) = \prod_{k=0}^{3} \epsilon_d^{(k)}$, where

$$\begin{aligned} \epsilon_a^{(k)} &= \mathrm{Cor}_3(\boldsymbol{\Phi}_k; \boldsymbol{\Phi}_k, \boldsymbol{\Theta}'_k), \quad & \epsilon_c^{(k)} &= \mathrm{Cor}_2(\boldsymbol{\Lambda}_k; \boldsymbol{\Lambda}_k, \boldsymbol{\Lambda}_k, \boldsymbol{\Theta}_k \oplus \boldsymbol{\Gamma}_k), \\ \epsilon_b^{(k)} &= \mathrm{Cor}_3(\boldsymbol{\Gamma}_k; \boldsymbol{\Gamma}_k, \boldsymbol{\Lambda}'_k), \quad & \epsilon_d^{(k)} &= \mathrm{Cor}_1(\boldsymbol{\Lambda}'_k; \boldsymbol{\Phi}_k), \end{aligned}$$

thus the correlation of the linear approximation relation (9) corresponding to the linear mask tuple $(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})$ can be computed as follows:

$$\epsilon_{\mathrm{FSM}}(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda}) = \prod_{k=0}^{3} \epsilon_b^{(k)} \epsilon_d^{(k)} \sum_{\boldsymbol{\Theta}} (\prod_{k=0}^{3} \epsilon_a^{(k)} \epsilon_c^{(k)}). \tag{10}$$

In the following part, we will show how to search for the mask tuples $(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})$ such that $|\epsilon_{\mathrm{FSM}}(\boldsymbol{\Phi}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda})|$ are as large as possible.

4.3 Search for Linear Masks $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ for $\epsilon_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$

In this part, we hope to find mask tuples $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ such that $|\epsilon_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})|$ computed by Equation (10) are as large as possible. Obviously, executing the search for all possible mask values is impractical. Therefore, we consider to use a search strategy attempting to find some potential linear masks.

For ease of description, we define the following two sets $\mathcal{S}_{id}$ and $\overline{\mathcal{S}}_{id}$ for $id \in \{0, 1, 2, 3\}$:

$$\mathcal{S}_{id} = \{\mathbf{\Lambda} = (\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3) : \mathbf{\Lambda}_{id} \in \mathbf{F}_{2^{32}}^*, \mathbf{\Lambda}_k = \mathbf{0} \in \mathbf{F}_{2^{32}} \text{ for all } k \neq id\},$$
$$\overline{\mathcal{S}}_{id} = \{\mathbf{\Lambda} = (\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3) : \mathbf{\Lambda}_{id} = \mathrm{0xa} \in \mathbf{F}_{2^8}^*, \mathbf{\Lambda}_k = \mathbf{0} \in \mathbf{F}_{2^{32}} \text{ for all } k \neq id\}.$$

Note that $\overline{\mathcal{S}}_{id} \subset \mathcal{S}_{id}$ for a fixed value id. For $\mathbf{\Lambda}$, let $\mathbf{\Lambda}'$ be the mask such that $\mathbf{\Lambda}' \cdot \mathbf{X} = \mathbf{\Lambda} \cdot \mathtt{MixColumns}(\mathtt{ShiftRows}(\mathbf{X}))$ for all $\mathbf{X} \in \mathbf{F}_{2^{128}}$. According to the computation of $\mathbf{\Lambda}'$ from $\mathbf{\Lambda}$ in Appendix D, we have the following observations:

(1) If $\mathbf{\Lambda} \in \mathcal{S}_{id}$ and $\mathbf{\Lambda}' \in \mathcal{S}_{id}$, then $\mathbf{\Lambda}' \in \overline{\mathcal{S}}_{id}$.
(2) If $\mathbf{\Lambda}' \in \overline{\mathcal{S}}_{id}$, then $\mathbf{\Lambda} \in \mathcal{S}_{id}$, and there are totally 255 choices of $\mathbf{\Lambda}_{id}$ for $\mathbf{\Lambda} \in \mathcal{S}_{id}$, which are listed in Table 3 of Appendix F.

Based on this, we propose the following search strategy. Intuitively, we hope most terms of the product in (10) to be 1. To achieve this, we choose the masks $\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}$ such that $\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda} \in \mathcal{S}_{id}$ for any given $id \in \{0, 1, 2, 3\}$. Also, we hope $\mathbf{\Lambda}' \in \mathcal{S}_{id}$. According to the above observations, we have $\mathbf{\Lambda}' \in \overline{\mathcal{S}}_{id}$ and obtain 255 possible values for $\mathbf{\Lambda}$. Besides, we can deduce according to Corollary 1 that the terms of the sum in Equation (10) have nonzero values if and only if $\mathbf{\Theta} \in \mathcal{S}_{id}$ and $\mathbf{\Theta}' \in \mathcal{S}_{id}$, thus $\mathbf{\Theta}' \in \overline{\mathcal{S}}_{id}$, and $\mathbf{\Theta}$ have the same 255 possible choices with $\mathbf{\Lambda}$. Under the above conditions, we derive

$$\epsilon_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}) = \epsilon_b^{(id)} \epsilon_d^{(id)} \sum_{\mathbf{\Theta}' \in \overline{\mathcal{S}}_{id}} \epsilon_a^{(id)} \epsilon_c^{(id)}. \tag{11}$$

Let id take a fixed value. Our search strategy for the mask tuples $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ begins by setting $\mathbf{\Phi} \in \mathcal{S}_{id}, \mathbf{\Gamma} \in \mathcal{S}_{id}$ and choosing $\mathbf{\Lambda} \in \mathcal{S}_{id}$ from Table 3. Then the search will be carried out according to the following steps.

Step 1: Consider the term $\epsilon_d^{(id)} = \mathrm{Cor}_1(\mathbf{\Lambda}'_{id}; \mathbf{\Phi}_{id})$ in Equation (11). Since $\mathbf{\Lambda}' \in \overline{\mathcal{S}}_{id}$, then $\mathbf{\Lambda}'_{id} \in \mathbf{F}_{2^8}^*$, we get that $\epsilon_d^{(id)} \neq 0$ only if $\mathbf{\Phi}_{id} \in \mathbf{F}_{2^8}^*$, i.e., $\mathbf{\Phi} \in \overline{\mathcal{S}}_{id}$. Thus, we pick out 255 "promising" values for $\mathbf{\Phi}$. Based on this, we tried all 255×255 combinations of the selected $(\mathbf{\Phi}, \mathbf{\Lambda}')$ and computed the values of $\epsilon_d^{(id)}$.

Step 2: Consider the term $\epsilon_b^{(id)} = \mathrm{Cor}_3(\mathbf{\Gamma}_{id}; \mathbf{\Gamma}_{id}, \mathbf{\Lambda}'_{id})$ in Equation (11). Note that $\mathbf{\Lambda}'_{id} \in \mathbf{F}_{2^8}^*$, according to Corollary 1, we deduce that $\epsilon_b^{(id)} \neq 0$ only if $\mathbf{\Gamma}_{id} \in \mathbf{F}_{2^8}^*$, i.e., $\mathbf{\Gamma} \in \overline{\mathcal{S}}_{id}$. Thus there are also 255 possibilities for $\mathbf{\Gamma}$. We tried all 255×255 combinations of the selected $(\mathbf{\Gamma}, \mathbf{\Lambda}')$ and computed the values of $\epsilon_b^{(id)}$.

Step 3: Consider the terms $\epsilon_a^{(id)} = \mathrm{Cor}_3(\mathbf{\Phi}_{id}; \mathbf{\Phi}_{id}, \mathbf{\Theta}'_{id})$ and $\epsilon_c^{(id)} = \mathrm{Cor}_2(\mathbf{\Lambda}_{id}; \mathbf{\Lambda}_{id}, \mathbf{\Lambda}_{id}, \mathbf{\Theta}_{id} \oplus \mathbf{\Gamma}_{id})$ in Equation (11). For all the $255 \times 255 \times 255 \approx 2^{24}$ choices of $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$, we can compute $\sum_{\mathbf{\Theta}' \in \overline{\mathcal{S}}_{id}} \epsilon_a^{(id)} \epsilon_c^{(id)}$ by including 255 terms of the sum over $\mathbf{\Theta}$.

Results. In summary, we have tried all 2^{24} combinations of $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ and computed the values of $\epsilon_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ by Equation (11). The total time complexity is $\mathcal{O}(2^{32})$. We have obtained some results, listed in Table 1. Our best results are the linear mask tuples $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ where $\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda} \in \mathcal{S}_{id}$ and $\mathbf{\Phi}_{id}$=0x00000002, $\mathbf{\Gamma}_{id}$=0x00000002, $\mathbf{\Lambda}_{id}$=0x0f0c0805. The corresponding best bitwise linear approximations have the correlation $\pm 2^{-18.67}$ and thus the SEI (squared correlation) $2^{-37.34}$.

Table 1: The partial linear mask tuples $(\mathbf{\Phi}_{id}, \mathbf{\Gamma}_{id}, \mathbf{\Lambda}_{id})$ of $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ for the bitwise linear approximations of the FSM of SNOW-V$_{\sigma_0}$

$\mathbf{\Phi}_{id}$	$\mathbf{\Gamma}_{id}$	$\mathbf{\Lambda}_{id}$	$\log(\lvert \sum \epsilon_a^{(id)} \epsilon_c^{(id)} \rvert)$	$\log(\lvert \epsilon_b^{(id)} \rvert)$	$\log(\lvert \epsilon_d^{(id)} \rvert)$	$\log(\lvert \epsilon_{\mathrm{FSM}} \rvert)$
0x00000002	0x00000002	0x0f0c0805	-11.71	-3.54	-3.42	$\mathbf{-18.67}$
0x00000007	0x00000002	0x0f0c0805	-11.77	-3.54	-3.42	-18.73
0x00000031	0x00000002	0x0f0c0805	-12.30	-3.54	-3.00	-18.84
0x00000003	0x00000002	0x0f0c0805	-11.71	-3.54	-3.68	-18.93

4.4 Using the Bitwise Linear Approximations in a Fast Correlation Attack on SNOW-V$_{\sigma_0}$

The bitwise linear approximations of the FSM of SNOW-V$_{\sigma_0}$ have the following form:

$$\mathbf{\Phi} \cdot z_{t-1} \oplus \mathbf{\Gamma} \cdot z_t \oplus \mathbf{\Lambda} \cdot z_{t+1} = \mathbf{\Phi} \cdot T1_{t-1} \oplus \mathbf{\Gamma} \cdot T1_t \oplus \mathbf{\Lambda} \cdot (T1_{t+1} \oplus T2_t) \oplus n^{(t)}.$$

We use the $4 \times 4 = 16$ mask tuples in Table 1 for approximations (id takes $0, 1, 2, 3$). These bitwise linear approximation relations have an average correlation of $\alpha \triangleq 2^{-18.79}$.

Generally, the fast correlation attack is a key recovery attack, trying to recover the key by utilizing the correlation between the keystream and the output of the LFSR states, which is commonly modeled as a decoding problem, as that done in [13, 14, 15, 21]. We need to decode a binary $[N, l]$-linear code through a Binary Symmetry Channel (BSC) with the error probability $p = \frac{1}{2}(1 - \alpha)$, where $\alpha = 2^{-18.79}$. The model is shown in Fig. 4.

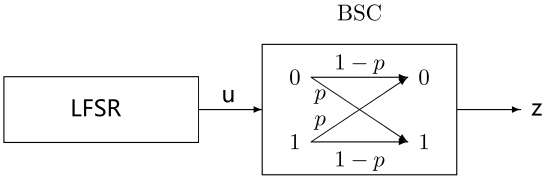

Figure 4: Model for a bitwise fast correlation attack

Accordingly, the bitwise fast correlation attack on SNOW-V$_{\sigma_0}$ is divided into the preprocessing phase and the processing phase. In the preprocessing phase, we first collect N samples involving only the keystream words and $l = 512$ LFSR initial state bits, and then try to reduce the number of the involved LFSR initial state bits to $l'(< l)$ bits at the expense of a folded noise level by searching for some 4-tuples from these samples which vanish on the most significant $l - l'$ bits to generate parity check equations. After this, we enter the processing phase to recover the target l' bits by using the Fast Walsh Transform (FWT) as that done in [5, 16], and further the whole LFSR initial state of SNOW-V$_{\sigma_0}$.

For example, if $l' = 244$, we need about $2l' \ln 2/(\alpha^4)^2 = 2^{158.71}$ samples with correlation α^4 to recover them, i.e., the number of 4-tuples found from N samples should be at least $2^{158.71}$. To ensure this, the number N should satisfy $\binom{N}{4} 2^{-(l-l')} \geq 2^{158.71}$, i.e., N should be at least $2^{107.83}$. We let $N = 2^{107.83}$, an approach to find all these sums requires a time complexity of $\mathcal{O}(N^2 \log N) = \mathcal{O}(2^{222.40})$. By using these $2^{158.71}$ new samples, we can recover 244 bits of the LFSR initial state using the FWT with a time complexity of $\mathcal{O}(2^{158.71} + l'2^{l'}) = \mathcal{O}(2^{251.93})$ and a memory complexity of $\mathcal{O}(2^{244})$. The above procedure requires a keystream of length $\frac{2^{107.83}}{16} = 2^{103.83}$. Though all the complexities are lower than 2^{256}, the keystream length of SNOW-V$_{\sigma_0}$ has been limited to a maximum of 2^{64} for a single pair of key and IV vectors. So this is not a feasible attack.

4.5 Comparison

In the design document of SNOW-V [8], the designers have devoted one section to SNOW-V$_{\sigma_0}$ (see Section 3.4 of [8]), where they construct the *byte-wise* linear approximations and compute the SEI of large distributions by the convolution algorithms, giving their best result with the SEI $2^{-58.7}$. With this byte-wise linear approximation, they give a fast correlation attack using the method in [23]. The time complexity is about 2^{232}, and the required length of the keystream is 2^{203}.

Different from the designer's method which utilizes large-unit approximations, this paper exploits bitwise linear approximations to mount the fast correlation attacks. In the above sections, we have described how to employ the linear-time algorithms in Section 3 to efficiently compute the correlations of the noise terms n_a, n_b, n_c and n_d, and finally obtain $\epsilon_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ for any given linear mask tuple $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ of SNOW-V$_{\sigma_0}$. All these algorithms combine the so-called "slice-like" technique, where some specific matrices independent of the given linear mask are pre-computed, and the correlation of the bitwise linear approximation for any given mask is computed by doing some matrix multiplications using the pre-computed matrices. Our search algorithms cost only linear-time complexities for an arbitrary given linear mask tuple, while the convolution algorithms on large distribution in [8] need much more computations. Based on these algorithms, we carry out a larger range of search for bitwise masks, and successfully found many stronger approximations which are "outside" the byte-wise approximations given in [8], thus resulting in better fast correlation attacks on SNOW-V$_{\sigma_0}$. The best bitwise linear approximations we found have the SEI (squared correlation) $2^{-37.34}$, while the best byte-wise linear approximation given in [8] has the SEI $2^{-58.7}$. That is, the new-found bitwise linear approximations have significantly larger SEI values than that of the best 8-bit linear approximation in [8]. Using the stronger approximations, we present a fast correlation attack, which costs a time complexity of $2^{251.93}$ and a memory complexity of 2^{244}, less than the exhaustive key search, and requires a keystream of length around $2^{103.83}$, much less than that provided in [8] which is 2^{203}.

5 Analysis of SNOW-V When Using σ as Proposed

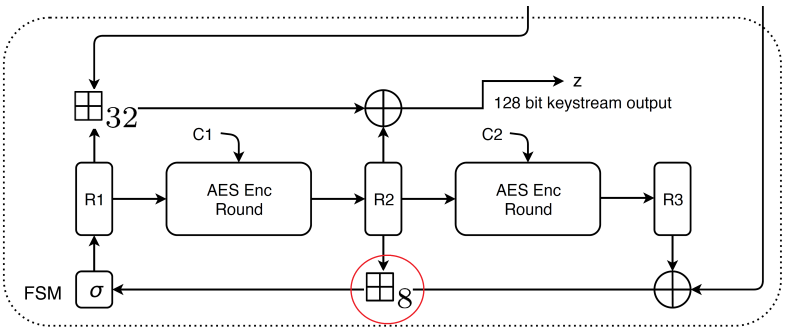

Figure 5: The FSM part of SNOW-V$_{\boxplus_{32},\boxplus_8}$

In this section, we will analyze the resistance of SNOW-V against linear approximation attacks when using σ as proposed. In the design document of SNOW-V [8], the authors give their research results on a variant of SNOW-V, where both of the two 32-bit adders "$\boxplus_{32}$" in the FSM are replaced by the 8-bit adders "$\boxplus_8$", which we denote by SNOW-V$_{\boxplus_8,\boxplus_8}$. Note that we will give our analysis on SNOW-V$_{\boxplus_8,\boxplus_8}$ in Section 6. Here, we focus on another variant of SNOW-V, where only the 32-bit adder used for updating the first register

$R1$ is replaced by "$\boxplus_8$", as depicted in Fig. 5, while everything else remains identical. We denote it SNOW-V$_{\boxplus_{32},\boxplus_8}$.

5.1 Bitwise Linear Approximation of the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$

For SNOW-V$_{\boxplus_{32},\boxplus_8}$, as shown in Fig. 5, the register $R1$ is updated according to $R1_{t+1} = \sigma((T2_t \oplus R3_t) \boxplus_8 R2_t)$. By applying the masks $\mathbf{\Phi}$, $\mathbf{\Gamma}$ and $\mathbf{\Lambda}$ to z_{t-1}, z_t and z_{t+1} respectively, we have

$$\begin{aligned}
\mathbf{\Phi} \cdot z_{t-1} &= \mathbf{\Phi} \cdot (T1_{t-1} \boxplus_{32} u_t) \oplus \mathbf{\Phi} \cdot v_t,\\
\mathbf{\Gamma} \cdot z_t &= \mathbf{\Gamma} \cdot (T1_t \boxplus_{32} w_t) \oplus \mathbf{\Gamma} \cdot AES^R(u_t),\\
\mathbf{\Lambda} \cdot z_{t+1} &= \mathbf{\Lambda} \cdot (T1_{t+1} \boxplus_{32} \sigma((T2_t \oplus AES^R(v_t)) \boxplus_8 AES^R(u_t))) \oplus \mathbf{\Lambda} \cdot AES^R(w_t),
\end{aligned}$$

where $u_t = R1_{t-1}$, $v_t = R2_{t-1}$ and $w_t = R1_t$. In this case, we consider the bitwise linear approximations with the following form:

$$\begin{aligned}
&(\boldsymbol{a}) \quad \mathbf{\Phi} \cdot (T1_{t-1} \boxplus_{32} u_t) = \mathbf{\Phi} \cdot T1_{t-1} \oplus \mathbf{\Theta} \cdot AES^R(u_t) \oplus n_a^{(t)},\\
&(\boldsymbol{b}) \quad \mathbf{\Gamma} \cdot (T1_t \boxplus_{32} w_t) = \mathbf{\Gamma} \cdot T1_t \oplus \mathbf{\Lambda} \cdot AES^R(w_t) \oplus n_b^{(t)},\\
&(\overline{\boldsymbol{c}}) \quad \mathbf{\Lambda} \cdot (T1_{t+1} \boxplus_{32} \sigma((T2_t \oplus AES^R(v_t)) \boxplus_8 AES^R(u_t)))\\
&\qquad = \mathbf{\Lambda} \cdot T1_{t+1} \oplus \sigma(\mathbf{\Lambda}) \cdot (T2_t \oplus AES^R(v_t)) \oplus (\mathbf{\Theta} \oplus \mathbf{\Gamma}) \cdot AES^R(u_t) \oplus n_{\bar{c}}^{(t)},\\
&(\overline{\boldsymbol{d}}) \quad \sigma(\mathbf{\Lambda}) \cdot AES^R(v_t) = \mathbf{\Phi} \cdot v_t \oplus n_{\bar{d}}^{(t)},
\end{aligned}$$

where $n_a^{(t)}$ and $n_b^{(t)}$ are the noises same as that in Section 4.1, and $n_{\bar{c}}^{(t)}$ and $n_{\bar{d}}^{(t)}$ are new introduced noises by new linear approximation relations $(\overline{\boldsymbol{c}})$ and $(\overline{\boldsymbol{d}})$ respectively. Now let $n^{(t)} = n_a^{(t)} \oplus n_b^{(t)} \oplus n_{\bar{c}}^{(t)} \oplus n_{\bar{d}}^{(t)}$, then the bitwise linear approximation of the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$ is

$$\mathbf{\Phi} \cdot z_{t-1} \oplus \mathbf{\Gamma} \cdot z_t \oplus \mathbf{\Lambda} \cdot z_{t+1} = \mathbf{\Phi} \cdot T1_{t-1} \oplus \mathbf{\Gamma} \cdot T1_t \oplus \mathbf{\Lambda} \cdot T1_{t+1} \oplus \sigma(\mathbf{\Lambda}) \cdot T2_t \oplus n^{(t)}. \quad (12)$$

Let $\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ denote the correlation of this linear approximation under the given mask tuple $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$. Then we have[2]

$$\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}) = \epsilon(n_b)\epsilon(n_{\bar{d}}) \sum\nolimits_{\mathbf{\Theta}} \epsilon(n_a)\epsilon(n_{\bar{c}}).$$

In the following, we will represent any 128-bit mask $\mathbf{\Lambda}$ in words as $\mathbf{\Lambda} = (\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3)$ with $\mathbf{\Lambda}_k \in \mathbf{F}_{2^{32}}$ for $k = 0, 1, 2, 3$, and $\mathbf{\Lambda}_k$ can be further expressed in bytes as $\mathbf{\Lambda}_k = (\mathbf{\Lambda}_{k,0} \parallel \mathbf{\Lambda}_{k,1} \parallel \mathbf{\Lambda}_{k,2} \parallel \mathbf{\Lambda}_{k,3})$ with $\mathbf{\Lambda}_{k,j} \in \mathbf{F}_{2^8}$ for $j = 0, 1, 2, 3$. In this case, $\mathbf{\Lambda}$ and $\sigma(\mathbf{\Lambda})$ are mapped to the following matrix expressions respectively:

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_{0,0} & \mathbf{\Lambda}_{0,1} & \mathbf{\Lambda}_{0,2} & \mathbf{\Lambda}_{0,3} \\ \mathbf{\Lambda}_{1,0} & \mathbf{\Lambda}_{1,1} & \mathbf{\Lambda}_{1,2} & \mathbf{\Lambda}_{1,3} \\ \mathbf{\Lambda}_{2,0} & \mathbf{\Lambda}_{2,1} & \mathbf{\Lambda}_{2,2} & \mathbf{\Lambda}_{2,3} \\ \mathbf{\Lambda}_{3,0} & \mathbf{\Lambda}_{3,1} & \mathbf{\Lambda}_{3,2} & \mathbf{\Lambda}_{3,3} \end{pmatrix}, \sigma(\mathbf{\Lambda}) = \begin{pmatrix} \mathbf{\Lambda}_{0,0} & \mathbf{\Lambda}_{1,0} & \mathbf{\Lambda}_{2,0} & \mathbf{\Lambda}_{3,0} \\ \mathbf{\Lambda}_{0,1} & \mathbf{\Lambda}_{1,1} & \mathbf{\Lambda}_{2,1} & \mathbf{\Lambda}_{3,1} \\ \mathbf{\Lambda}_{0,2} & \mathbf{\Lambda}_{1,2} & \mathbf{\Lambda}_{2,2} & \mathbf{\Lambda}_{3,2} \\ \mathbf{\Lambda}_{0,3} & \mathbf{\Lambda}_{1,3} & \mathbf{\Lambda}_{2,3} & \mathbf{\Lambda}_{3,3} \end{pmatrix}, \mathbf{\Lambda}_{k,j} \in \mathbf{F}_{2^8}$$

5.2 Search for Linear Masks $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ for $\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$

5.2.1 Computation of the Correlations of n_a and n_b

Note that $n_a^{(t)}$ and $n_b^{(t)}$ are the same as that in Section 4.1, which can be rewritten as follows:

$$\begin{aligned}
n_a^{(t)} &= \mathbf{\Phi} \cdot (T1_{t-1} \boxplus_{32} \mathtt{SubBytes}^{-1}(x_t)) \oplus \mathbf{\Phi} \cdot T1_{t-1} \oplus \mathbf{\Theta}' \cdot x_t,\\
n_b^{(t)} &= \mathbf{\Gamma} \cdot (T1_t \boxplus_{32} \mathtt{SubBytes}^{-1}(y_t)) \oplus \mathbf{\Gamma} \cdot T1_t \oplus \mathbf{\Lambda}' \cdot y_t.
\end{aligned}$$

[2] We simplify the noises $n_a^{(t)}$, $n_b^{(t)}$, $n_{\bar{c}}^{(t)}$ and $n_{\bar{d}}^{(t)}$ by n_a, n_b, $n_{\bar{c}}$ and $n_{\bar{d}}$ respectively.

where $\epsilon(n_a) = \prod_{k=0}^{3} \mathrm{Cor}_3(\mathbf{\Phi}_k; \mathbf{\Phi}_k, \mathbf{\Theta}'_k)$ and $\epsilon(n_b) = \prod_{k=0}^{3} \mathrm{Cor}_3(\mathbf{\Gamma}_k; \mathbf{\Gamma}_k, \mathbf{\Lambda}'_k)$.

Let $\mathcal{R}_{k,j} = (\mathbf{\Phi}_{k,j}, \mathbf{\Phi}_{k,j}, \mathbf{\Theta}'_{k,j})$ for $k = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$. According to Theorem 3 in Appendix C, we have

$$\epsilon(n_a) = \prod_{k=0}^{3} \mathrm{Cor}_3(\mathbf{\Phi}_k; \mathbf{\Phi}_k, \mathbf{\Theta}'_k) = \prod_{k=0}^{3} \boldsymbol{l}_2 \mathbf{M}_{\mathcal{R}_{k,3}} \mathbf{M}_{\mathcal{R}_{k,2}} \mathbf{M}_{\mathcal{R}_{k,1}} \mathbf{M}_{\mathcal{R}_{k,0}} \boldsymbol{e}_0.$$

Similarly, let $\mathcal{R}'_{k,j} = (\mathbf{\Gamma}_{k,j}, \mathbf{\Gamma}_{k,j}, \mathbf{\Lambda}'_{k,j})$ for $k = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$, we have

$$\epsilon(n_b) = \prod_{k=0}^{3} \mathrm{Cor}_3(\mathbf{\Gamma}_k; \mathbf{\Gamma}_k, \mathbf{\Lambda}'_k) = \prod_{k=0}^{3} \boldsymbol{l}_2 \mathbf{M}_{\mathcal{R}'_{k,3}} \mathbf{M}_{\mathcal{R}'_{k,2}} \mathbf{M}_{\mathcal{R}'_{k,1}} \mathbf{M}_{\mathcal{R}'_{k,0}} \boldsymbol{e}_0.$$

5.2.2 Computation of the Correlation of $n_{\bar{c}}$

For the bitwise linear approximation relation $(\overline{\boldsymbol{c}})$, let $\xi_t = T2_t \oplus AES^R(v_t)$ and $\eta_t = AES^R(u_t)$, then the noise $n_{\bar{c}}^{(t)}$ can be expressed as follows:

$$n_{\bar{c}}^{(t)} = \mathbf{\Lambda} \cdot (T1_{t+1} \boxplus_{32} \sigma(\xi_t \boxplus_8 \eta_t)) \oplus \mathbf{\Lambda} \cdot T1_{t+1} \oplus \sigma(\mathbf{\Lambda}) \cdot \xi_t \oplus (\mathbf{\Theta} \oplus \mathbf{\Gamma}) \cdot \eta_t.$$

Obviously, $n_{\bar{c}}$ can be viewed as the noise introduced by the bitwise linear approximation of the **Type-V** function $\mathcal{H}$ defined in Section 3.1 with the mask tuple $(\mathbf{\Lambda}; \mathbf{\Lambda}, \sigma(\mathbf{\Lambda}), \mathbf{\Theta} \oplus \mathbf{\Gamma})$. For $\mathcal{H}$, we have the parameters $l = 128$, $n = 32$, $m = 8$ and the permutation $\sigma = [0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15]$, equivalent to the transposition of a 4×4 matrix.

Let $\mathcal{V}_{k,j} = (\mathbf{\Lambda}_{k,j}, \mathbf{\Lambda}_{k,j}, \mathbf{\Lambda}_{k,j}, \mathbf{\Theta}_{j,k} \oplus \mathbf{\Gamma}_{j,k})$ for $k = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$. According to Equation (8) in Section 3.3 for computing the bitwise linear approximation of $\mathcal{H}$, we have

$$\epsilon(n_{\bar{c}}) = \mathrm{Cor}_5(\mathbf{\Lambda}; \mathbf{\Lambda}, \sigma(\mathbf{\Lambda}), \mathbf{\Theta} \oplus \mathbf{\Gamma}) = \prod_{k=0}^{3} \boldsymbol{l}_2 \mathbf{C}_{\mathcal{V}_{k,3}} \mathbf{C}_{\mathcal{V}_{k,2}} \mathbf{C}_{\mathcal{V}_{k,1}} \mathbf{C}_{\mathcal{V}_{k,0}} \boldsymbol{e}_0.$$

Complexity. As shown in Section 3.3, to compute the correlation of $n_{\bar{c}}$ with the mask tuple $(\mathbf{\Lambda}; \mathbf{\Lambda}, \sigma(\mathbf{\Lambda}), \mathbf{\Theta} \oplus \mathbf{\Gamma})$, we need to pre-compute the matrices $\mathbf{C}_{\mathcal{V}}$ for all $\mathcal{V}$ in the form of $\mathcal{V} = (\gamma^{(0)}, \gamma^{(0)}, \gamma^{(0)}, \gamma^{(1)})$, where $\gamma^{(0)}, \gamma^{(1)} \in \mathbf{F}_{2^8}$. For each $\mathcal{V}$ of this form, $\mathbf{C}_{\mathcal{V}}$ can be pre-computed by Theorem 1 by doing 8 matrix multiplications of small size, and thus all the matrices $\mathbf{C}_{\mathcal{V}}$ in the form of $\mathcal{V} = (\gamma^{(0)}, \gamma^{(0)}, \gamma^{(0)}, \gamma^{(1)})$ can be obtained by doing $2^{16} \times 8$ matrix multiplications of small size. After all these matrices are pre-computed, we can compute $\epsilon(n_{\bar{c}}) = \mathrm{Cor}_5(\mathbf{\Lambda}; \mathbf{\Lambda}, \sigma(\mathbf{\Lambda}), \mathbf{\Theta} \oplus \mathbf{\Gamma})$ for the fixed mask tuple $(\mathbf{\Lambda}, \mathbf{\Lambda}, \sigma(\mathbf{\Lambda}), \mathbf{\Theta} \oplus \mathbf{\Gamma})$ by doing 16 matrix multiplications of small size.

5.2.3 Computation of the Correlation of $n_{\bar{d}}$

For the bitwise linear approximation relation $(\overline{\boldsymbol{d}})$, we denote $\sigma(\mathbf{\Lambda})'$ the mask such that $\sigma(\mathbf{\Lambda})' \cdot \mathbf{X} = \sigma(\mathbf{\Lambda}) \cdot \mathtt{MixColumns}(\mathtt{ShiftRows}(\mathbf{X}))$ for all $\mathbf{X} \in \mathbf{F}_{2^{128}}$ (See Appendix D for the computation of $\sigma(\mathbf{\Lambda})'$ from $\sigma(\mathbf{\Lambda})$), then we have $\sigma(\mathbf{\Lambda}) \cdot AES^R(v_t) = \sigma(\mathbf{\Lambda})' \cdot \mathtt{SubBytes}(v_t)$, and thus the noise $n_{\bar{d}}^{(t)}$ can be expressed as

$$n_{\bar{d}}^{(t)} = \sigma(\mathbf{\Lambda})' \cdot \mathtt{SubBytes}(v_t) \oplus \mathbf{\Phi} \cdot v_t,$$

where $\mathtt{SubBytes}(\cdot)$ is an application of 16 AES S-boxes S_R. We use the notation $\widehat{\mathbf{\Lambda}}$ to denote $\sigma(\mathbf{\Lambda})'$, then the correlation of $n_{\bar{d}}$ can be computed as

$$\epsilon(n_{\bar{d}}) = \prod_{k=0}^{3} \epsilon_{S_R}(\widehat{\mathbf{\Lambda}}_{k,3}; \mathbf{\Phi}_{k,3}) \epsilon_{S_R}(\widehat{\mathbf{\Lambda}}_{k,2}; \mathbf{\Phi}_{k,2}) \epsilon_{S_R}(\widehat{\mathbf{\Lambda}}_{k,1}; \mathbf{\Phi}_{k,1}) \epsilon_{S_R}(\widehat{\mathbf{\Lambda}}_{k,0}; \mathbf{\Phi}_{k,0}),$$

which can be obtained by looking up the LAT pre-computed in Section 4.2.3 16 times.

5.2.4 Search for Linear Masks $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$

For a given mask tuple $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$, the correlation of the linear approximation of the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$ is

$$\bar{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}) = \epsilon(n_b)\epsilon(n_{\bar{d}}) \sum\nolimits_{\mathbf{\Theta}} \epsilon(n_a)\epsilon(n_{\bar{c}}). \tag{13}$$

We will use the notation $\mathcal{S}_{id}$ defined in Section 4.3 to illustrate our search strategy, where $\mathcal{S}_{id} = \{\mathbf{\Lambda} = (\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3) : \mathbf{\Lambda}_{id} \in \mathbf{F}_{2^{32}}^*, \mathbf{\Lambda}_k = \mathbf{0} \in \mathbf{F}_{2^{32}} \text{ for all } k \neq id\}$. In our attempt to search for $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ such that the linear approximation of the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$ would be highly biased, we observed that when both $\mathbf{\Phi} \in \mathcal{S}_{id}$ and $\mathbf{\Gamma} \in \mathcal{S}_{id}$ are satisfied, $|\bar{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})|$ is more likely to have high value. According to Corollary 1, we must have $\mathbf{\Theta}' \in \mathcal{S}_{id}$ (Resp. $\mathbf{\Lambda}' \in \mathcal{S}_{id}$) to guarantee $\epsilon(n_a) \neq 0$ (Resp. $\epsilon(n_b) \neq 0$). Besides, considering the term $\epsilon(n_{\bar{d}})$, since $\mathbf{\Phi} \in \mathcal{S}_{id}$, we get that $\epsilon(n_{\bar{d}}) \neq 0$ only if $\sigma(\mathbf{\Lambda})' \in \mathcal{S}_{id}$. Thus, the constraints for $\mathbf{\Lambda}$ is both $\mathbf{\Lambda}' \in \mathcal{S}_{id}$ and $\sigma(\mathbf{\Lambda})' \in \mathcal{S}_{id}$ are satisfied, from which we derive that $\sigma(\mathbf{\Lambda}) = \mathbf{\Lambda}$, and for each fixed value of $id \in \{0, 1, 2, 3\}$, there are 255 choices for $\mathbf{\Lambda}$, which are defined as follows.

Choices for $\mathbf{\Lambda}$. Let id take a fixed value. We take one of the 255 elements from Table 3 and assign it to $\mathbf{\Lambda}_{id}$, and define $\mathbf{\Lambda}_{(id+1) mod 4} = \mathbf{\Lambda}_{id} \ggg_{32} 8$, $\mathbf{\Lambda}_{(id+2) mod 4} = \mathbf{\Lambda}_{id} \ggg_{32} 16$ and $\mathbf{\Lambda}_{(id+3) mod 4} = \mathbf{\Lambda}_{id} \ggg_{32} 24$, then $\mathbf{\Lambda} = (\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3)$ is one of the 255 candidates for $\mathbf{\Lambda}$. For example, let $id = 0$, we take the element `0x0f0c0805` from Table 3 and assign it to $\mathbf{\Lambda}_0$, and define $\mathbf{\Lambda}_1$ =`0x050f0c08`, $\mathbf{\Lambda}_2$ =`0x08050f0c` and $\mathbf{\Lambda}_3$ =`0x0c08050f`, then the following $\mathbf{\Lambda}$ is one of the 255 candidates, satisfying $\sigma(\mathbf{\Lambda}) = \mathbf{\Lambda}$ and $\sigma(\mathbf{\Lambda})' = \mathbf{\Lambda}' \in \mathcal{S}_{id}$ (here $id = 0$), i.e.,

$$\begin{aligned} \mathbf{\Lambda} &= \texttt{0x0c08050f08050f0c050f0c080f0c0805} = \sigma(\mathbf{\Lambda}) \\ \mathbf{\Lambda}' &= \texttt{0x00000000000000000000000000e0e0e0e} \in \mathcal{S}_0. \end{aligned}$$

Further, taking into consideration the term $\sum_{\mathbf{\Theta}} \epsilon(n_a)\epsilon(n_{\bar{c}})$, we have the following observation from many examples of our experiments though we do not have a proof:

- For the linear masks $\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}$ and $\mathbf{\Theta}$ such that: *(1)* $\mathbf{\Phi} \in \mathcal{S}_{id}$; *(2)* $\mathbf{\Theta}' \in \mathcal{S}_{id}$; *(3)* $\mathbf{\Gamma} \in \mathcal{S}_{id}$; *(4)* $\sigma(\mathbf{\Lambda}) = \mathbf{\Lambda}$ and $\sigma(\mathbf{\Lambda})' = \mathbf{\Lambda}' \in \mathcal{S}_{id}$ (totally 255 candidates for $\mathbf{\Lambda}$), we observed that the value of $\epsilon(n_a)\epsilon(n_{\bar{c}})$ with $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$ is nearly equal to the value of $\sum_{\mathbf{\Theta}} \epsilon(n_a)\epsilon(n_{\bar{c}})$, that is to say, taking just one term with $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$ in Equation (13) yields sufficiently accurate estimates of the total correlation, i.e.,

$$\bar{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}) \approx \epsilon(n_a)\epsilon(n_b)\epsilon(n_{\bar{c}})\epsilon(n_{\bar{d}}), \text{ where } \mathbf{\Theta} = \sigma(\mathbf{\Lambda}) = \mathbf{\Lambda}. \tag{14}$$

Based on the above observation, our search strategy for $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ is as follows.

Step 1: Let $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$, and consider the term $\epsilon(n_{\bar{c}})$ with $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$ in Equation (14). For each of the 255 candidates for $\mathbf{\Lambda}$, we exhaust all 2^{32} possible values of $\mathbf{\Gamma} \in \mathcal{S}_{id}$, and collect those $\mathbf{\Gamma}$ such that $|\epsilon(n_{\bar{c}})| \geq 2^{-46}$. By the experiments, we observed that only when

$$\begin{aligned} \mathbf{\Lambda} &= \texttt{0x0c08050f08050f0c050f0c080f0c0805}\ (id = 0),\ \textit{or} \\ \mathbf{\Lambda} &= \texttt{0x08050f0c050f0c080f0c08050c08050f}\ (id = 1),\ \textit{or} \\ \mathbf{\Lambda} &= \texttt{0x050f0c080f0c08050c08050f08050f0c}\ (id = 2),\ \textit{or} \\ \mathbf{\Lambda} &= \texttt{0x0f0c08050c08050f08050f0c050f0c08}\ (id = 3), \end{aligned}$$

there exists $\mathbf{\Gamma} \in \mathcal{S}_{id}$ such that $\epsilon(n_{\bar{c}}) \geq 2^{-50}$. For each one, we obtained 1596 such values for $\mathbf{\Gamma}$.

Step 2: Let $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$, and consider the term $\epsilon(n_b)\epsilon(n_{\bar{c}})$ in Equation (14). For $\mathbf{\Lambda} =$ `0x0c08050f08050f0c050f0c080f0c0805` $(id = 0)$, we try all the 1596 values for $\mathbf{\Gamma}$ obtained above, and collect those satisfying $|\epsilon(n_b)\epsilon(n_{\bar{c}})| \geq 2^{-62}$. We picked out 14 such values for $\mathbf{\Gamma}$.

Step 3: Let $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$, and consider the term $\epsilon(n_a)\epsilon(n_{\bar{d}})$ in Equation (14). Let $\mathbf{\Lambda}$ =0x0c08 050f08050f0c050f0c080f0c0805 ($id = 0$), we exhaust all 2^{32} possible values for $\mathbf{\Phi} \in \mathcal{S}_{id}$, and collect those $\mathbf{\Phi}$ satisfying $|\epsilon(n_a)\epsilon(n_{\bar{d}})| \geq 2^{-31}$. We obtained 199 such values for $\mathbf{\Phi}$.

Step 4: Let $\mathbf{\Lambda}$ = 0x0c08050f08050f0c050f0c080f0c0805 ($id = 0$), for all the 14×199 combinations of $(\mathbf{\Phi}, \mathbf{\Gamma})$ obtained above, we compute the accurate values of $\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ according to Equation (13) by including all 2^{32} terms of the sum over $\mathbf{\Theta}$ with $\mathbf{\Theta}' \in \mathcal{S}_{id}$, i.e., $\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}) = \epsilon(n_b)\epsilon(n_{\bar{d}}) \sum_{\mathbf{\Theta}' \in \mathcal{S}_{id}} \epsilon(n_a)\epsilon(n_{\bar{c}})$.

Results. In our best attempt to approximate the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$, we have obtained the total noise having the correlation $\pm 2^{-91.60}$, and thus the SEI $2^{-183.20}$. The best linear mask tuples $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ we found out are listed in Table 2. Besides, for $\mathbf{\Lambda}$ = 0x0c08050f08050f0c050f0c080f0c0805 ($id = 0$), we obtained 216 combinations of $(\mathbf{\Phi}, \mathbf{\Gamma})$ such that $|\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})| \geq 2^{-92}$, thus there are totally $4 \times 216 = 864$ mask tuples $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ (id takes $0, 1, 2, 3$), under which the SEI of the bitwise linear approximation of the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$ are larger than 2^{184}. Our results further confirm the observation made in [8] that the introduction of the permutation σ in the FSM part makes the bias of linear approximations for the FSM to be much smaller.

Table 2: The best linear masks $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ for the bitwise linear approximation of the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$

$\mathbf{\Lambda}$ ($id = 0$)	$\mathbf{\Phi} \in \mathcal{S}_0$	$\mathbf{\Gamma} \in \mathcal{S}_0$	$\log(\lvert\overline{\epsilon}_{\mathrm{FSM}}\rvert)$
0x0c08050f08050f0c 050f0c080f0c0805	0x0000000000000000 0000000007b32407	0x0000000000000000 0000000005000500	−91.603
0x0c08050f08050f0c 050f0c080f0c0805	0x0000000000000000 0000000002b32407	0x0000000000000000 0000000005000500	−91.606
$\mathbf{\Lambda}$ ($id = 1$)	$\mathbf{\Phi} \in \mathcal{S}_1$	$\mathbf{\Gamma} \in \mathcal{S}_1$	$\log(\lvert\overline{\epsilon}_{\mathrm{FSM}}\rvert)$
0x08050f0c050f0c08 0f0c08050c08050f	0x0000000000000000 07b3240700000000	0x0000000000000000 0500050000000000	−91.603
0x08050f0c050f0c08 0f0c08050c08050f	0x0000000000000000 02b3240700000000	0x0000000000000000 0500050000000000	−91.606
$\mathbf{\Lambda}$ ($id = 2$)	$\mathbf{\Phi} \in \mathcal{S}_2$	$\mathbf{\Gamma} \in \mathcal{S}_2$	$\log(\lvert\overline{\epsilon}_{\mathrm{FSM}}\rvert)$
0x050f0c080f0c0805 0c08050f08050f0c	0x0000000007b32407 0000000000000000	0x0000000005000500 0000000000000000	−91.603
0x050f0c080f0c0805 0c08050f08050f0c	0x0000000002b32407 0000000000000000	0x0000000005000500 0000000000000000	−91.606
$\mathbf{\Lambda}$ ($id = 3$)	$\mathbf{\Phi} \in \mathcal{S}_3$	$\mathbf{\Gamma} \in \mathcal{S}_3$	$\log(\lvert\overline{\epsilon}_{\mathrm{FSM}}\rvert)$
0x0f0c08050c08050f 08050f0c050f0c08	0x07b3240700000000 0000000000000000	0x0500050000000000 0000000000000000	−91.603
0x0f0c08050c08050f 08050f0c050f0c08	0x02b3240700000000 0000000000000000	0x0500050000000000 0000000000000000	−91.606

Remark. For the mask tuple $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ in the first row of Table 2 with $id = 0$, we computed by experiments the values of $\epsilon(n_a)\epsilon(n_{\bar{c}})$ for all $\mathbf{\Theta}$ such that $\mathbf{\Theta}' \in \mathcal{S}_{id}$ and thus obtained the accurate value of $\sum_{\mathbf{\Theta}' \in \mathcal{S}_{id}} \epsilon(n_a)\epsilon(n_{\bar{c}})$. Our results show that $\epsilon(n_a)\epsilon(n_{\bar{c}}) \neq 0$ only when $\mathbf{\Theta}$ takes the value of $\sigma(\mathbf{\Lambda})$, i.e., $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$. Thus the value of $\epsilon(n_a)\epsilon(n_{\bar{c}})$ with $\mathbf{\Theta} = \sigma(\mathbf{\Lambda})$ is actually equal to the value of $\sum_{\mathbf{\Theta}' \in \mathcal{S}_{id}} \epsilon(n_a)\epsilon(n_{\bar{c}}) (= \sum_{\mathbf{\Theta}} \epsilon(n_a)\epsilon(n_{\bar{c}}))$, which is an example to give an illustration of the above observation.

5.3 A Fast Correlation Attack on SNOW-V$_{\boxplus_{32},\boxplus_8}$

The bitwise linear approximations of the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$ have the following form:

$$\mathbf{\Phi} \cdot z_{t-1} \oplus \mathbf{\Gamma} \cdot z_t \oplus \mathbf{\Lambda} \cdot z_{t+1} = \mathbf{\Phi} \cdot T1_{t-1} \oplus \mathbf{\Gamma} \cdot T1_t \oplus \mathbf{\Lambda} \cdot T1_{t+1} \oplus \sigma(\mathbf{\Lambda}) \cdot T2_t \oplus n^{(t)}.$$

We use those 864 mask tuples $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ such that $|\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})| \geq 2^{-92}$ for approximations. The corresponding bitwise linear approximation relations have the correlation $|\overline{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})| \geq 2^{-92} \triangleq \alpha$.

We first collect N samples involving only the keystream words and the LFSR initial state bits of length $l = 512$, and then try to reduce the number of the involved LFSR initial state bits to $l'(< l)$ bits by searching for some pairs from the samples which vanish on the most significant $l - l'$ bits, at the expense of the increased noise level with the correlation α^2. There are about $M \triangleq N(N-1)2^{-(l-l'+1)}$ such pairs, corresponding to M approximation relations with correlation α^2 involving only l' bits of the LFSR initial state, which can be found by the sort-and-merge procedure with the time/memory complexity $\mathcal{O}(N)$. To recover the value of the target l' bits, we still use the FWT to speed up the evaluation of the M linear approximation relations, which needs a time complexity $\mathcal{O}(M + l'2^{l'})$ and a memory complexity $\mathcal{O}(2^{l'})$. Set $M = 2l' \ln 2/(\alpha^2)^2$, the parameter N is determined to be $N \approx \sqrt{M2^{l-l'+1}}$, and the required number of keystream outputs is $D = N/864$.

Complexity Analysis. For SNOW-V$_{\boxplus_{32},\boxplus_8}$, we follow the above procedure with the parameters $l = 512$, $l' = 363$. In this case, we need to prepare $M = 2l' \ln 2/(\alpha^2)^2 = 2^{376.98}$ approximation relations with correlation α^2 involving the first 363 bits of the LFSR initial state. The required number of samples is $N = \sqrt{M2^{l-l'+1}} = 2^{263.49}$ and we need to know $D = N/864 = 2^{253.73}$ keystream outputs. The required time/memory complexity for preparing M approximation relations is $2^{263.49}$. The FWT is utilized to determine the first 363 bits of the LFSR initial state, which needs a time complexity $2^{377.01}$ and a memory complexity 2^{363}. Once the first 363 bits are recovered, the other bits and the FSM state can be recovered by using a similar method and a small-scale exhaustive search with a much lower complexity.

6 Analysis of SNOW-V$_{\boxplus_8,\boxplus_8}$

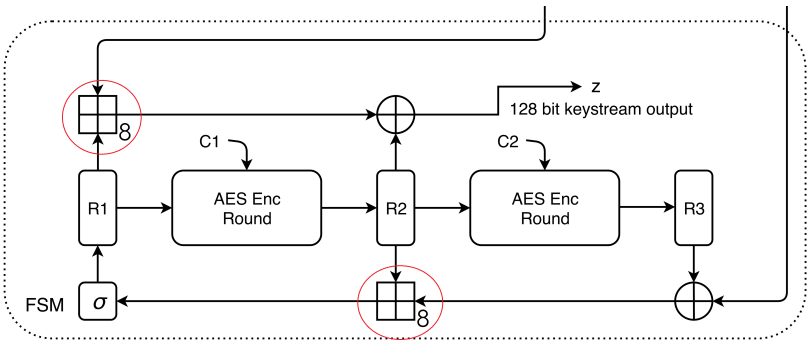

Figure 6: The FSM part of SNOW-V$_{\boxplus_8,\boxplus_8}$

In this section, we give a brief study on the bitwise linear approximation of the FSM of SNOW-V$_{\boxplus_8,\boxplus_8}$, of which the byte-wise linear approximation has been studied in the design document of SNOW-V [8]. With this simplification all operations are byte-oriented.

6.1 Bitwise Linear Approximation of the FSM of SNOW-V$_{\boxplus_8,\boxplus_8}$

For SNOW-V$_{\boxplus_8,\boxplus_8}$, we consider the bitwise linear approximations of the following form:

$$\begin{aligned}
n_1^{(t)} &= \mathbf{\Phi} \cdot (T1_{t-1} \boxplus_8 \texttt{SubBytes}^{-1}(x_t)) \oplus \mathbf{\Phi} \cdot T1_{t-1} \oplus \mathbf{\Theta}' \cdot x_t,\\
n_2^{(t)} &= \mathbf{\Gamma} \cdot (T1_t \boxplus_8 \texttt{SubBytes}^{-1}(y_t)) \oplus \mathbf{\Gamma} \cdot T1_t \oplus \mathbf{\Lambda}' \cdot y_t,\\
n_3^{(t)} &= \mathbf{\Lambda} \cdot (T1_{t+1} \boxplus_8 \sigma(\xi_t \boxplus_8 \eta_t)) \oplus \mathbf{\Lambda} \cdot T1_{t+1} \oplus \sigma(\mathbf{\Lambda}) \cdot \xi_t \oplus (\mathbf{\Theta} \oplus \mathbf{\Gamma}) \cdot \eta_t,\\
n_4^{(t)} &= \sigma(\mathbf{\Lambda})' \cdot \texttt{SubBytes}(v_t) \oplus \mathbf{\Phi} \cdot v_t.
\end{aligned}$$

In this case, the bitwise linear approximation of the FSM of SNOW-V$_{\boxplus_8,\boxplus_8}$ is

$$\mathbf{\Phi} \cdot z_{t-1} \oplus \mathbf{\Gamma} \cdot z_t \oplus \mathbf{\Lambda} \cdot z_{t+1} = \mathbf{\Phi} \cdot T1_{t-1} \oplus \mathbf{\Gamma} \cdot T1_t \oplus \mathbf{\Lambda} \cdot T1_{t+1} \oplus \sigma(\mathbf{\Lambda}) \cdot T2_t \oplus n^{(t)}.$$

where $n^{(t)} = n_1^{(t)} \oplus n_2^{(t)} \oplus n_3^{(t)} \oplus n_4^{(t)}$. Let $\widetilde{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$ denote the correlation of this linear approximation under a given mask tuple $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})$. Then we have

$$\widetilde{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda}) = \epsilon(n_2)\epsilon(n_4) \sum\nolimits_{\mathbf{\Theta}} \epsilon(n_1)\epsilon(n_3).$$

Computation of the Correlations of $\mathbf{n_1}$ and $\mathbf{n_2}$. Let $\mathcal{R}_{k,j} = (\mathbf{\Phi}_{k,j}, \mathbf{\Phi}_{k,j}, \mathbf{\Theta}'_{k,j})$ and $\mathcal{R}'_{k,j} = (\mathbf{\Gamma}_{k,j}, \mathbf{\Gamma}_{k,j}, \mathbf{\Lambda}'_{k,j})$ for $k = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$, it's easy to deduce that

$$\epsilon(n_1) = \prod_{k=0}^{3} \prod_{j=0}^{3} \boldsymbol{l}_2 \mathbf{M}_{\mathcal{R}_{k,j}} \boldsymbol{e}_0, \ \epsilon(n_2) = \prod_{k=0}^{3} \prod_{j=0}^{3} \boldsymbol{l}_2 \mathbf{M}_{\mathcal{R}'_{k,j}} \boldsymbol{e}_0.$$

Computation of the Correlation of $\mathbf{n_3}$. Let $\mathcal{V}_{k,j} = (\mathbf{\Lambda}_{k,j}, \mathbf{\Lambda}_{k,j}, \mathbf{\Lambda}_{k,j}, \mathbf{\Theta}_{j,k} \oplus \mathbf{\Gamma}_{j,k})$ for $k = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$, we deduce that

$$\epsilon(n_3) = \prod_{k=0}^{3} \prod_{j=0}^{3} \boldsymbol{l}_2 \mathbf{C}_{\mathcal{V}_{k,j}} \boldsymbol{e}_0.$$

Computation of the Correlation of $\mathbf{n_4}$. With the notations as before, we have

$$\epsilon(n_4) = \prod_{k=0}^{3} \prod_{j=0}^{3} \epsilon_{S_R}(\widehat{\mathbf{\Lambda}}_{k,j}; \mathbf{\Phi}_{k,j}).$$

Results. We will sketch some ideas on how to find good linear approximations, but list the best linear approximations we found for SNOW-V$_{\boxplus_8,\boxplus_8}$. The results when using 8-bit adders to replace 32-bit adders are as follows ($id = 0$):

$$\begin{aligned}
\mathbf{\Lambda} &= \texttt{0x0c08050f08050f0c050f0c080f0c0805},\\
\mathbf{\Gamma} &= \texttt{0x00000000000000000000000001040202},\\
\mathbf{\Phi} &= \texttt{0x00000000000000000000000002020202},
\end{aligned}$$

such that

$$|\epsilon(n_2)| = 2^{-19.50}, \ |\epsilon(n_4)| = 2^{-13.66}, \ |\sum\nolimits_{\mathbf{\Theta}} \epsilon(n_1)\epsilon(n_3)| = 2^{-53.91}, \ |\widetilde{\epsilon}_{\mathrm{FSM}}(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{\Lambda})| = 2^{-87.07}.$$

6.2 Comparison

As explained in Section 4.5, the analysis from the original design document of SNOW-V [8] relied on byte-wise linear approximations, where the best byte-wise approximation for the FSM of SNOW-V$_{\boxplus_8,\boxplus_8}$ has the SEI $2^{-214.80}$, whereas ours uses bitwise linear approximations, via the techniques from Section 3. We have computed the correlations of

the involved noise terms n_1, n_2, n_3 and n_4 by doing some matrix multiplications using the pre-computed matrices in Section 3, which cost only linear-time complexities and thus allows for a wide range of search for bitwise masks. In our experiment, we have found a number of stronger bitwise approximations than the best byte-wise one in [8], among which the best bitwise one has the SEI $2^{-174.14}$. Thus we have increased the bias of the linear approximation from $2^{-214.80}$ to $2^{-174.14}$, which is a big improvement.

7 Conclusion

In this paper, we present a number of stronger linear approximations for the FSM of several variants of SNOW-V, i.e., SNOW-V$_{\sigma_0}$, SNOW-V$_{\boxplus_8,\boxplus_8}$ and SNOW-V$_{\boxplus_{32},\boxplus_8}$, and further propose attacks accordingly, resulting in the bitwise fast correlation attacks faster than those in the design document of SNOW-V [8]. We first propose and summarize some efficient algorithms using the slice-like techniques to compute the linear approximations of certain types of composition functions composed of basic operations like $\boxplus$, $\oplus$, *Permutation* and *S-box*, which are the underlying functions arising in the linear approximations of SNOW-like stream ciphers. Based on this, we find some bitwise linear approximations for the FSM of SNOW-V$_{\sigma_0}$ with the SEI around $2^{-37.34}$ and mount a bitwise fast correlation attack with the time complexity $2^{251.93}$ and memory complexity 2^{244}, given $2^{103.83}$ keystream outputs, which improves greatly the results in the design document. Besides, we find our best bitwise linear approximations for the FSM of SNOW-V$_{\boxplus_8,\boxplus_8}$ with the SEI $2^{-174.14}$, while the best byte-wise linear approximation in [8] has the SEI $2^{-214.80}$. Further, we study a closer variant of SNOW-V, i.e., SNOW-V$_{\boxplus_{32},\boxplus_8}$, we derive many mask tuples for the FSM of SNOW-V$_{\boxplus_{32},\boxplus_8}$, yielding linear approximations with the SEI larger than 2^{-184}. Using these linear approximations, we mount a fast correlation attack with the time complexity $2^{377.01}$ and a memory complexity 2^{363}, given $2^{253.73}$ keystream outputs. Although neither of our attacks threatens the security of SNOW-V, we provide new lights on the structure of SNOW-like stream ciphers and also the bitwise linear approximation attacks. We think the research in this paper is meaningful for our future work to study the bitwise linear approximation of SNOW-V and mount attacks accordingly.

Further discussion. We first make a brief discussion on how likely similar attacks would be on full SNOW-V. For full SNOW-V, two 32-bit adders "$\boxplus_{32}$" are used in the FSM part. One is used to generate the 128-bit keystream as $z_t = (T1_t \boxplus_{32} R1_t) \oplus R2_t$, and the other is used to update the first register $R1$ as $R1_{t+1} = \sigma((T2_t \oplus R3_t) \boxplus_{32} R2_t)$. In the course of approximating the FSM, we can derive a new type of function $\mathcal{G} : \mathbf{F}_{2^{128}} \times \mathbf{F}_{2^{128}} \times \mathbf{F}_{2^{128}} \to \mathbf{F}_{2^{128}}$ such that $\mathcal{G}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}) = \mathbf{X}^{(1)} \boxplus_{32} \sigma(\mathbf{X}^{(2)} \boxplus_{32} \mathbf{X}^{(3)})$. Bitwise linear approximations for the FSM of SNOW-V can be constructed if the bitwise linear approximation of $\mathcal{G}$ could be efficiently computed, and thus similar attacks would be mounted. Actually, we have been doing research on full SNOW-V and achieved some initial results. It is one of our future work to study deeply the bitwise linear approximations of SNOW-V. Beside, it is well known that the SEI of a bitwise linear approximation is always smaller than or equal to the SEI of a multidimensional linear approximation that covers the masks of the bitwise approximations. We will also study the large-unit linear approximations of SNOW-V or SNOW-V variants in the future.

Acknowledgments

We would like to thank all reviewers for providing valuable comments to the manuscript. This work is supported by the National Key R&D Research programm (Grant No. 2017YFB0802504), the program of the National Natural Science Foundation of China (Grant No. 61572482), National Cryptography Development Fund (Grant No.

MMJJ20170107).

References

[1] Baigneres T., Junod P., Vaudenay S.: How far can we go beyond linear cryptanalysis? *Advances in Cryptology-ASIACRYPT 2004*, LNCS vol. 3329, pp. 113-128, Springer Berlin Heidelberg (2004).

[2] Berbain C., Billet O., Canteaut A., Courtois N., Gilbert H., Goubin L., Gouget A., Granboulan L., Lauradoux C., Minier M., Pornin T., Sibert H.: Sosemanuk, a fast software-oriented stream cipher, In: Robshaw M., Billet O. (eds.), *New Stream Cipher Designs*, LNCS vol. 4986, pp. 98-118, Springer Berlin Heidelberg (2008).

[3] Coppersmith D., Halevi S., Jutla C.: Cryptanalysis of stream ciphers with linear masking. In Yung M. (ed.), *Advances in Cryptology-CRYPTO 2002*, LNCS vol. 2442, pp. 515-532, Springer Berlin Heidelberg (2002).

[4] Chepyzhov V. V., Johansson T., Smeets B.: A simple algorithm for fast correlation attacks on stream ciphers. In: Goos G., Hartmanis J., Leeuwen J. V., Schneier B. (eds.), *Fast Software Encryption-FSE 2001*, LNCS vol. 1978, pp. 181-195, Springer Berlin Heidelberg (2001).

[5] Chose P., Joux A., Mitton M.: Fast correlation attacks: an algorithmic point of view. In: Knudsen L. R. (ed.), *Advances in Cryptology-EUROCRYPT 2002*, LNCS vol. 2332, pp. 209-221, Springer Berlin Heidelberg (2002).

[6] Ekdahl P., Johansson T.: A new version of the stream cipher SNOW. In: Nyberg K., Heys H. (eds.), *Selected Areas in Cryptography-SAC 2002*, LNCS vol. 2595, pp. 47-61, Springer Berlin Heidelberg (2003).

[7] Ekdahl P., Johansson T.: SNOW-a new stream cipher. *Proceedings of First Open NESSIE Workshop*, pp. 167-168 (2000).

[8] Ekdahl P., Johansson T., Maximov A., Yang J.: A new SNOW stream cipher called SNOW-V. *IACR Trans. Symmetric Cryptol.* vol. 2019(3), pp. 1-42 (2019).

[9] ETSI/SAGE: Specification of the 3gpp confidentiality and integrity algorithms UEA2 & UIA2. *Document 2: SNOW 3G Specification, version 1.1*, `http://www.3gpp.org/ftp/`, September 2006.

[10] Funabiki Y., Todo Y., Isobe T., Morii M.: Several MILP-aided attacks against SNOW 2.0. In: Camenisch J., Papadimitratos P. (eds.), *Cryptology and Network Security-CANS 2018*, LNCS vol. 11124, pp. 394-413, Springer Berlin Heidelberg (2018).

[11] Gong X., Zhang B.: Fast computation of linear approximation over certain composition functions and applications to SNOW 2.0 and SNOW 3G. *Des. Codes Cryptogr*, LNCS vol. 88, pp. 2407-2431, `https://doi.org/10.1007/s10623-020-00790-3` (2020).

[12] Jiao L., Li Y., Hao Y.: A guess-and-determine attack on SNOW-V stream cipher. *Comput. J.*. LNCS vol. 63, pp. 1789-181, `https://doi.org/10.1093/comjnl/bxaa003` (2020).

[13] Johansson T., Jönsson F.: Improved fast correlation attacks on stream ciphers via convolutional codes. In: Stern J. (ed.), *Advances in Cryptology-EUROCRYPT 1999*, LNCS vol. 1592, pp. 347-362, Springer Berlin Heidelberg (1999).

[14] Johansson T., Jönsson F.: Fast correlation attacks through reconstruction of linear polynomials. In: Bellare M. (ed.), *Advances in Cryptology-CRYPTO 2000*, LNCS vol. 1880, pp. 300-315, Springer Berlin Heidelberg (2000).

[15] Lee J. K., Lee D. H., Park S.: Cryptanalysis of Sosemanuk and SNOW 2.0 using linear masks. In: Pieprzyk J. (ed.), *Advances in Cryptology-ASIACRYPT 2008*, LNCS vol. 5350, pp. 524-538, Springer Berlin Heidelberg (2008).

[16] Lu Y., Vaudenay S.: Faster correlation attack on bluetooth keystream generator E0. In: Franklin M. (ed.), *Advances in Cryptology-CRYPTO 2004*, LNCS vol. 3152, pp. 407-425, Springer Berlin Heidelberg (2004).

[17] Matsui M.: Linear cryptanalysis method for DES cipher. *Advances in Cryptology-EUROCRYPT 1993*, LNCS vol. 765, pp. 386-397 (1993).

[18] Nyberg K.: Correlation theorems in cryptanalysis. *Discrete Applied Mathematics*, vol. 111, pp. 177-188 (2001).

[19] Nyberg K., Wallén J.: Improved linear distinguishers for SNOW 2.0. In: Robshaw M. (ed.), *Fast Software Encryption-FSE 2006*, LNCS vol. 4047, pp. 144-162, Springer Berlin Heidelberg (2006).

[20] Watanabe D., Biryukov A., De Cannière C.: A Distinguishing attack of SNOW 2.0 with linear masking method. In: Matsui M., Zuccherato R. J. (eds.), *Selected Areas in Cryptography-SAC 2003*, LNCS vol. 3006, pp. 222-233, Springer Berlin Heidelberg (2004).

[21] Zhang B., Gong X., Meier W.: Fast correlation attacks on Grain-like small state stream ciphers. *IACR Transactions on Symmetric Cryptology*, ISSN 2519-173X, vol. 2017(4), pp. 58-81. doi:10.13154/tosc.v2017.i4.58-81

[22] Yang J., Johansson T., Maximov A.: Vectorized linear approximations for attacks on SNOW 3G. *IACR Transactions on Symmetric Cryptology*, ISSN, 2519-173X, vol. 2019(4), pp. 249–171. doi:10.13154/tosc.v2019.i4.249-271

[23] Zhang B., Xu C., Meier W.: Fast correlation attacks over extension fields, large-unit linear approximation and cryptanalysis of SNOW 2.0. In: Gennaro R., Robshaw M. (eds.), *Advances in Cryptology-CRYPTO 2015*, LNCS vol. 9215, pp. 643-662, Springer Berlin Heidelberg (2015).

A Computing the Bitwise Linear Approximation of Type-I Function

Problem 1. Compute the correlation of the bitwise linear approximation of the Type-I function $Sbox(\cdot)$ with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input mask $\mathbf{\Gamma}^{(1)}$, which is denoted by $\mathrm{Cor}_1(\mathbf{\Gamma}^{(0)};\mathbf{\Gamma}^{(1)})$.

Method and Complexity. Since $Sbox(\cdot)$ is composed of parallel operations of s_j, for a given mask tuple $(\mathbf{\Gamma}^{(0)},\mathbf{\Gamma}^{(1)})$, we write $\mathbf{\Gamma}^{(0)} = (\mathbf{\Gamma}_0^{(0)} \parallel \mathbf{\Gamma}_1^{(0)} \parallel ... \parallel \mathbf{\Gamma}_{d-1}^{(0)})$ and $\mathbf{\Gamma}^{(1)} = (\mathbf{\Gamma}_0^{(1)} \parallel \mathbf{\Gamma}_1^{(1)} \parallel ... \parallel \mathbf{\Gamma}_{d-1}^{(1)})$ with $\mathbf{\Gamma}_j^{(0)}, \mathbf{\Gamma}_j^{(1)} \in \mathbf{F}_{2^m}$ for $j = 0, 1, ..., d-1$. According to the Piling-up lemma, we have

$$\mathrm{Cor}_1(\mathbf{\Gamma}^{(0)};\mathbf{\Gamma}^{(1)}) = \prod_{j=0}^{d-1} \epsilon_{\mathrm{s}_j}(\mathbf{\Gamma}_j^{(0)};\mathbf{\Gamma}_j^{(1)}).$$

The computation is usually carried out according to the following two phases:

- In the preprocessing phase, we pre-compute d tables to store all the linear approximations of s_j for $j = 0, 1, ..., d-1$. Since the most common size of s_j are 4-bit or 8-bit, we use the trivial way to compute $\epsilon_{\mathrm{s}_j}(\gamma^{(0)}; \gamma^{(1)})$ by trying all possibilities of $\gamma^{(0)}$, $\gamma^{(1)}$ values. For this, for each $\gamma^{(0)} \in \mathbf{F}_{2^m}$ and $\gamma^{(1)} \in \mathbf{F}_{2^m}$, we loop for each $\mathbf{x} \in \mathbf{F}_{2^m}$ to compute the values of $\gamma^{(0)} \cdot \mathrm{s}_j(\mathbf{x}) \oplus \gamma^{(1)} \cdot \mathbf{x}$, and can finally derive $\epsilon_{\mathrm{s}_j}(\gamma^{(0)}; \gamma^{(1)})$ for all $j = 0, 1, ..., d-1$ with a total time complexity of $\mathcal{O}(d2^{3m})$, affordable for small m. All the values $\epsilon_{\mathrm{s}_j}(\gamma^{(0)}; \gamma^{(1)})$ are stored in the row indexed by $(\gamma^{(0)}, \gamma^{(1)})$ of the j-th table. Thus the total memory complexity is $\mathcal{O}(d2^{2m})$.
- In the processing phase, the correlation of the linear approximation of $Sbox(\cdot)$ for each given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)})$ can be derived by table lookups d times indexed by $(\mathbf{\Gamma}_j^{(0)}; \mathbf{\Gamma}_j^{(1)})$ in the j-th table, respectively. This is a *linear-time* procedure.

B Computing the Bitwise Linear Approximation of Type-II Function

Problem 2. Compute the correlation of the bitwise linear approximation of the Type-II function F with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks $\mathbf{\Gamma}^{(1)}, ..., \mathbf{\Gamma}^{(\rho)}$, which is denoted by $\mathrm{Cor}_2(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, ..., \mathbf{\Gamma}^{(\rho)})$.

Method and Complexity. In [19], the authors have proposed a linear-time algorithm to compute the correlation of the bitwise linear approximation of F for any given mask tuple, we describe it in the following theorem.

Theorem 2. **[19]**. *Let $\rho > 1$ be a fixed integer. For each $\mathcal{R} = (r_0, r_1, ..., r_\rho) \in \mathbf{F}_2^{\rho+1}$, let $\mathbf{D}_\mathcal{R}$ be the $\rho \times \rho$ matrix with the (oc, ic)-element for $ic, oc \in \{0, ..., \rho-1\}$ computed as*

$$\begin{aligned}\mathbf{D}_\mathcal{R}[oc][ic] = 2^{-k}(&|\{\mathbf{x} \in \mathbf{F}_2^\rho :\ r_0 \cdot f(\mathbf{x}, ic) = \bar{\mathbf{r}} \cdot \mathbf{x},\ g(\mathbf{x}, ic) = oc\}| \\ &-|\{\mathbf{x} \in \mathbf{F}_2^\rho :\ r_0 \cdot f(\mathbf{x}, ic) \neq \bar{\mathbf{r}} \cdot \mathbf{x},\ g(\mathbf{x}, ic) = oc\}|),\end{aligned}$$

where[3]

$$\begin{aligned}&\bar{\mathbf{r}} = (r_1, ..., r_\rho) \in \mathbf{F}_2^\rho, \quad \mathbf{x} = (x_1, ..., x_\rho) \in \mathbf{F}_2^\rho, \\ &f(\mathbf{x}, ic) = (w_H(\mathbf{x}) + ic) \bmod 2, \quad g(\mathbf{x}, ic) = \lfloor (w_H(\mathbf{x}) + ic)/2 \rfloor .\end{aligned}$$

Let $\boldsymbol{l}_\rho$ be the row vector of length ρ with all elements equal to 1, and let $\boldsymbol{e}_0$ be the column vector of length ρ with a single 1 in 0-th row and zero otherwise. For any given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, ..., \mathbf{\Gamma}^{(\rho)})$ of the ρ-input addition modulo 2^n, write $\mathbf{\Gamma}^{(i)}$ in bits as $\mathbf{\Gamma}^{(i)} = (\gamma_0^{(i)} \parallel \gamma_1^{(i)} \parallel ... \parallel \gamma_{n-1}^{(i)})$, $i = 0, 1, ..., \rho$, then we have

$$\mathrm{Cor}_2(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, ..., \mathbf{\Gamma}^{(\rho)}) = \boldsymbol{l}_\rho \mathbf{D}_{\mathcal{R}_{n-1}} ... \mathbf{D}_{\mathcal{R}_1} \mathbf{D}_{\mathcal{R}_0} \boldsymbol{e}_0,$$

where $\mathcal{R}_j = (\gamma_j^{(0)}, \gamma_j^{(1)}, ..., \gamma_j^{(\rho)}) \in \mathbf{F}_2^{\rho+1}$ for $j = 0, 1, ..., n-1$.

According to Theorem 2, the procedure for computing the correlation of the bitwise linear approximation of F for any given mask tuple can be divided into the following two phases:

- In the preprocessing phase, $2^{\rho+1}$ matrices of size $\rho \times \rho$ should be pre-computed and stored. For each given $\mathcal{R} \in \mathbf{F}_2^{\rho+1}$, the matrix $\mathbf{D}_\mathcal{R}$ can be constructed with a time complexity of $\mathcal{O}(\rho 2^\rho)$ and a memory complexity of $\mathcal{O}(\rho^2)$, thus it requires a total time complexity of $\mathcal{O}(\rho 2^{2\rho+1})$ and a memory complexity of $\mathcal{O}(\rho^2 2^{\rho+1})$ to pre-compute $\mathbf{D}_\mathcal{R}$ for all the possibilities of $\mathcal{R}$.

[3] Note that $w_H(\mathbf{x})$: $0 \leq w_H(\mathbf{x}) \leq \rho$, denotes the Hamming weight of $\mathbf{x}$, which is the number of non-zero components of $\mathbf{x}$.

- In the processing phase, the correlation of the linear approximation of F for each given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, ..., \mathbf{\Gamma}^{(\rho)})$ can be obtained by doing n multiplications of a $\rho \times \rho$ matrix and a column vector, and n additional additions, which is a *linear-time* algorithm for a fixed ρ.

C Computing the Bitwise Linear Approximation of Type-III Function

Problem 3. Compute the correlation of the bitwise linear approximation of the Type-III function G with respect to the output mask $\mathbf{\Gamma}^{(0)}$ and the input masks $\mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}$, which is denoted by $\mathrm{Cor}_3(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)})$.

Method and Complexity. In [11], a linear-time algorithm is proposed to compute the correlation of the linear approximation of G under any given mask tuple, and then used to mount attacks on SNOW 2.0 and SNOW 3G. At a very high level, the idea is to divide the n-bit values into d values of m-bit according to the specific structure of the function $Sbox(\cdot)$, and then pre-compute and store some useful matrices, and finally compute the correlation by doing some matrix multiplications using these pre-computed matrices. We now describe this method in short.

According to the structure of $Sbox(\cdot)$, the masks $\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}$ and $\mathbf{\Gamma}^{(2)}$ are split into d blocks and each block has m bits, i.e., $\mathbf{\Gamma}^{(i)} = (\mathbf{\Gamma}_0^{(i)} \parallel \mathbf{\Gamma}_1^{(i)} \parallel ... \parallel \mathbf{\Gamma}_{d-1}^{(i)})$, where $\mathbf{\Gamma}_j^{(i)} \in \mathbf{F}_{2^m}$ for $i = 0, 1, 2$ and $j = 0, 1, ..., d-1$. Similarly, we also split the variables $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ into d blocks of m-bit as $\mathbf{x}^{(0)} = (\mathbf{x}_0^{(0)} \parallel \mathbf{x}_1^{(0)} \parallel ... \parallel \mathbf{x}_{d-1}^{(0)})$, $\mathbf{x}^{(1)} = (\mathbf{x}_0^{(1)} \parallel \mathbf{x}_1^{(1)} \parallel ... \parallel \mathbf{x}_{d-1}^{(1)})$. Let $u_j = \mathbf{x}_j^{(1)} + \mathrm{s}_j(\mathbf{x}_j^{(2)})$, we define the Boolean functions g_j for $j = 0, 1, ..., d-1$ as follows:

$$g_j(\theta_j, \mathbf{x}_j^{(1)}, \mathbf{x}_j^{(2)}) = \mathbf{\Gamma}_j^{(0)} \cdot (u_j \boxplus_m \theta_j) \oplus \mathbf{\Gamma}_j^{(1)} \cdot \mathbf{x}_j^{(1)} \oplus \mathbf{\Gamma}_j^{(2)} \cdot \mathbf{x}_j^{(2)},$$

where $\theta_0 = 0$ and $\theta_{j+1} = \lfloor (u_j + \theta_j)/2^m \rfloor$. Based on the above, we summarize the method in [11] in the following theorem, which provides an efficient method to compute the bitwise linear approximation of G.

Algorithm 1 Construction of the matrix $\mathbf{M}_{(j,(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)}))}$

Parameters: the parallel (m, m)-functions s_j, $j \in \{0, 1, ..., d-1\}$
Notations: $+$, the usual integer addition;
$\lfloor\ \rfloor$, the floor function of integers;
mod, the modulo operation of integers;
Input: the partial mask values $\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}$ with $\gamma^{(i)} \in \mathbf{F}_{2^m}$, $i = 0, 1, 2$

1: Create a matrix $\mathbf{M}_{(j,(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)}))}$ of size 2×2;
2: Create two 2×2 matrices $\mathbf{N}_0$ and $\mathbf{N}_1$ initialized with zeros;
3: **for** $i\theta \in \{0, 1\}$, $\mathbf{x}^{(1)} \in \mathbf{F}_{2^m}$ and $\mathbf{x}^{(2)} \in \mathbf{F}_{2^m}$ **do**
4: compute $u = \mathbf{x}^{(1)} + \mathrm{s}_j(\mathbf{x}^{(2)})$;
5: compute $r = \gamma^{(0)} \cdot ((u + i\theta) \bmod 2^m) \oplus \gamma^{(1)} \cdot \mathbf{x}^{(1)} \oplus \gamma^{(2)} \cdot \mathbf{x}^{(2)}$;
6: compute $o\theta = \lfloor (u + i\theta)/2^m \rfloor$;
7: $\mathbf{N}_r[o\theta][i\theta] := \mathbf{N}_r[o\theta][i\theta] + 1$;
8: **for** $i\theta \in \{0, 1\}$ and $o\theta \in \{0, 1\}$ **do**
9: $\mathbf{M}_{(j,(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)}))}[o\theta][i\theta] := (\mathbf{N}_0[o\theta][i\theta] - \mathbf{N}_1[o\theta][i\theta])/2^{2m}$;

Output: the corresponding 2×2 matrix $\mathbf{M}_{(j,(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)}))}$

Theorem 3. [11]. *Let $\boldsymbol{l}_2 = (1, 1)$ be a row vector and $\boldsymbol{e}_0 = (1, 0)^{\mathrm{T}}$ be a column vector. For any given mask tuple $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)})$, we have*

$$\mathrm{Cor}_3(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)}) = \boldsymbol{l}_2 \mathbf{M}_{(d-1,\mathcal{R}_{d-1})} ... \mathbf{M}_{(1,\mathcal{R}_1)} \mathbf{M}_{(0,\mathcal{R}_0)} \boldsymbol{e}_0,$$

where $\mathcal{R}_0, \mathcal{R}_1, ..., \mathcal{R}_{d-1}$ *are vectors defined according to* $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)})$ *as* $\mathcal{R}_j = (\mathbf{\Gamma}_j^{(0)}, \mathbf{\Gamma}_j^{(1)}, \mathbf{\Gamma}_j^{(2)})$*, and* $\mathbf{M}_{(j,\mathcal{R}_j)}$ *are* 2×2 *matrices pre-computed by* Algorithm 1 *such that*

$$\begin{aligned}\mathbf{M}_{(j,\mathcal{R}_j)}[\alpha][\theta_j] = 2^{-2m}(&|\{\mathbf{x}_j^{(1)}, \mathbf{x}_j^{(2)} \in \mathbf{F}_{2^m} : g_j(\theta_j, \cdot) = 0,\ \theta_{j+1}(\theta_j, \cdot) = \alpha\}| \\ &- |\{\mathbf{x}_j^{(1)}, \mathbf{x}_j^{(2)} \in \mathbf{F}_{2^m} : g_j(\theta_j, \cdot) = 1,\ \theta_{j+1}(\theta_j, \cdot) = \alpha\}|),\end{aligned}$$

for $\alpha \in \{0, 1\}$ *and* $\theta_j \in \{0, 1\}$

Complexity Analysis. From Algorithm 1, for each given partial mask values $\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}$ and a given function s_j, the matrix $\mathbf{M}_{(j,(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)}))}$ can be constructed with a time complexity $\mathcal{O}(2^{2m+1})$ and a memory complexity $\mathcal{O}(1)$. To search for all those mask tuples $(\mathbf{\Gamma}^{(0)}, \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)})$ such that $|\mathrm{Cor}_3(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)})|$ are as large as possible, we seem to need to compute $\mathbf{M}_{(j,(\gamma^{(0)},\gamma^{(1)},\gamma^{(2)}))}$ for all the possible values of $\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}$ and at most d different functions s_j, and thus $d2^{3m}$ matrices should be pre-computed, which requires a pre-computation complexity of $\mathcal{O}(d2^{5m+1})$ and a memory complexity of $\mathcal{O}(d2^{3m})$ in the worst case. However, as shown (by experimental results) in [11], this number can be decreased by a factor of at least 2^m, since the correlation value $|\mathrm{Cor}_3(\mathbf{\Gamma}^{(0)}; \mathbf{\Gamma}^{(1)}, \mathbf{\Gamma}^{(2)})|$ may be larger with $\mathbf{\Gamma}^{(1)} = \mathbf{\Gamma}^{(0)}$ than with $\mathbf{\Gamma}^{(1)} \neq \mathbf{\Gamma}^{(0)}$, especially in case of SNOW-like ciphers. Using the pre-computed matrices, the correlation of the linear approximation of G for any given mask tuple can be accurately computed by doing d multiplications of small size, which costs a *linear-time* complexity of $\mathcal{O}(d)$, and thus allows for a wide range of search of highly biased linear approximations.

D Computing the mask Θ' from the given mask Θ such that $\Theta' \cdot x = \Theta \cdot \mathtt{MixColumns}(\mathtt{ShiftRows}(x))$

In the FSM of SNOW-V, the full AES round function (denoted by AES^R) is used to update the 128-bit registers. Given a 128-bit output mask $\mathbf{\Theta}$ of AES^R, we let $\mathbf{\Delta}$ be the mask computed from $\mathbf{\Theta}$ by combining the `MixColumns` of the AES round function such that

$$\mathbf{\Delta} \cdot y = \mathbf{\Theta} \cdot \mathtt{MixColumns}(y),\ \text{for all } y \in \mathbf{F}_{2^{128}},$$

and let $\mathbf{\Theta}'$ be the mask computed from $\mathbf{\Delta}$ by combining the `ShiftRows` of the AES round function such that

$$\mathbf{\Theta}' \cdot x = \mathbf{\Delta} \cdot \mathtt{ShiftRows}(x),\ \text{for all } x \in \mathbf{F}_{2^{128}}.$$

We then have

$$\mathbf{\Theta}' \cdot x = \mathbf{\Theta} \cdot \mathtt{MixColumns}(\mathtt{ShiftRows}(x)),\ \text{for all } x \in \mathbf{F}_{2^{128}}.$$

Before we describe how to compute the masks $\mathbf{\Delta}$ and $\mathbf{\Theta}'$, let us define a linear transformation $lin : \mathbf{F}_{2^{32}} \to \mathbf{F}_{2^{32}}$, where $\mathbf{\Lambda}' = lin(\mathbf{\Lambda})$ represents the linear mask computed from $\mathbf{\Lambda}$ by combining the MixColumn matrix (denoted by M) of the AES round function, i.e., $\mathbf{\Lambda}' \cdot x = \mathbf{\Lambda} \cdot Mx$ for all $x \in \mathbf{F}_{2^{32}}$.

We now briefly describe how to compute $\mathbf{\Lambda}'$ from $\mathbf{\Lambda}$. For $\lambda = (\lambda_0 \parallel \lambda_1 \parallel \lambda_2 \parallel \lambda_3 \parallel \lambda_4 \parallel \lambda_5 \parallel \lambda_6 \parallel \lambda_7) \in \mathbf{F}_{2^8}$, define $\lambda' = trans(\lambda) = (\lambda'_0 \parallel \lambda'_1 \parallel \lambda'_2 \parallel \lambda'_3 \parallel \lambda'_4 \parallel \lambda'_5 \parallel \lambda'_6 \parallel \lambda'_7)$ as:

$$\begin{cases} \lambda'_i = \lambda_{i+1}, \quad i = 0, 1, ..., 6 \\ \lambda'_7 = \underline{\lambda_0 \oplus \lambda_1 \oplus \lambda_3 \oplus \lambda_4}. \end{cases}$$

Write $\mathbf{\Lambda} = (\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3)$, $\mathbf{\Lambda}' = (\mathbf{\Lambda}'_0 \parallel \mathbf{\Lambda}'_1 \parallel \mathbf{\Lambda}'_2 \parallel \mathbf{\Lambda}'_3)$, where $\mathbf{\Lambda}_j \in \mathbf{F}_{2^8}$, $\mathbf{\Lambda}'_j \in \mathbf{F}_{2^8}$, we have $(\mathbf{\Lambda}'_0 \parallel \mathbf{\Lambda}'_1 \parallel \mathbf{\Lambda}'_2 \parallel \mathbf{\Lambda}'_3) = lin(\mathbf{\Lambda}_0 \parallel \mathbf{\Lambda}_1 \parallel \mathbf{\Lambda}_2 \parallel \mathbf{\Lambda}_3)$ such that

$$\begin{aligned}
\mathbf{\Lambda}'_0 &= trans(\mathbf{\Lambda}_0) \oplus \mathbf{\Lambda}_1 \oplus \mathbf{\Lambda}_2 \oplus \mathbf{\Lambda}_3 \oplus trans(\mathbf{\Lambda}_3).\\
\mathbf{\Lambda}'_1 &= trans(\mathbf{\Lambda}_1) \oplus \mathbf{\Lambda}_2 \oplus \mathbf{\Lambda}_3 \oplus \mathbf{\Lambda}_0 \oplus trans(\mathbf{\Lambda}_0),\\
\mathbf{\Lambda}'_2 &= trans(\mathbf{\Lambda}_2) \oplus \mathbf{\Lambda}_3 \oplus \mathbf{\Lambda}_0 \oplus \mathbf{\Lambda}_1 \oplus trans(\mathbf{\Lambda}_1),\\
\mathbf{\Lambda}'_3 &= trans(\mathbf{\Lambda}_3) \oplus \mathbf{\Lambda}_0 \oplus \mathbf{\Lambda}_1 \oplus \mathbf{\Lambda}_2 \oplus trans(\mathbf{\Lambda}_2).
\end{aligned}$$

Based on the above, we describe how to compute the masks $\mathbf{\Delta}$ and $\mathbf{\Theta}'$ respectively.

— **Computation of $\mathbf{\Delta}$.** For any $\mathbf{\Theta} \in \mathbf{F}_{2^{128}}$, we can rewrite it in the form of array as

$$\mathbf{\Theta} = \begin{pmatrix} \mathbf{\Theta}_0 \parallel \mathbf{\Theta}_1 \parallel \mathbf{\Theta}_2 \parallel \mathbf{\Theta}_3 \\ \mathbf{\Theta}_4 \parallel \mathbf{\Theta}_5 \parallel \mathbf{\Theta}_6 \parallel \mathbf{\Theta}_7 \\ \mathbf{\Theta}_8 \parallel \mathbf{\Theta}_9 \parallel \mathbf{\Theta}_{10} \parallel \mathbf{\Theta}_{11} \\ \mathbf{\Theta}_{12} \parallel \mathbf{\Theta}_{13} \parallel \mathbf{\Theta}_{14} \parallel \mathbf{\Theta}_{15} \end{pmatrix}, \text{ with } \mathbf{\Theta}_j \in \mathbf{F}_{2^8}, \text{ for } j = 0, 1, ..., 15$$

where $\mathbf{\Theta}_0$ is the least significant byte, and $\mathbf{\Theta}_{15}$ is the most significant byte. The mask $\mathbf{\Delta}$ is computed from $\mathbf{\Theta}$ by combining the `MixColumns` of the AES round function. which can be computed as follows,

$$\mathbf{\Delta} \triangleq \begin{pmatrix} \mathbf{\Delta}_0 \parallel \mathbf{\Delta}_1 \parallel \mathbf{\Delta}_2 \parallel \mathbf{\Delta}_3 \\ \mathbf{\Delta}_4 \parallel \mathbf{\Delta}_5 \parallel \mathbf{\Delta}_6 \parallel \mathbf{\Delta}_7 \\ \mathbf{\Delta}_8 \parallel \mathbf{\Delta}_9 \parallel \mathbf{\Delta}_{10} \parallel \mathbf{\Delta}_{11} \\ \mathbf{\Delta}_{12} \parallel \mathbf{\Delta}_{13} \parallel \mathbf{\Delta}_{14} \parallel \mathbf{\Delta}_{15} \end{pmatrix} = \begin{pmatrix} lin(\mathbf{\Theta}_0 \parallel \mathbf{\Theta}_1 \parallel \mathbf{\Theta}_2 \parallel \mathbf{\Theta}_3) \\ lin(\mathbf{\Theta}_4 \parallel \mathbf{\Theta}_5 \parallel \mathbf{\Theta}_6 \parallel \mathbf{\Theta}_7) \\ lin(\mathbf{\Theta}_8 \parallel \mathbf{\Theta}_9 \parallel \mathbf{\Theta}_{10} \parallel \mathbf{\Theta}_{11}) \\ lin(\mathbf{\Theta}_{12} \parallel \mathbf{\Theta}_{13} \parallel \mathbf{\Theta}_{14} \parallel \Lambda_{15}) \end{pmatrix},$$

— **Computation of $\mathbf{\Theta}'$.** The mask $\mathbf{\Theta}'$ is computed from $\mathbf{\Delta}$ by combining the `ShiftRows` of the AES round function, which can be computed as follows,

$$\mathbf{\Theta}' \triangleq \begin{pmatrix} \mathbf{\Theta}'_0 \parallel \mathbf{\Theta}'_1 \parallel \mathbf{\Theta}'_2 \parallel \mathbf{\Theta}'_3 \\ \mathbf{\Theta}'_4 \parallel \mathbf{\Theta}'_5 \parallel \mathbf{\Theta}'_6 \parallel \mathbf{\Theta}'_7 \\ \mathbf{\Theta}'_8 \parallel \mathbf{\Theta}'_9 \parallel \mathbf{\Theta}'_{10} \parallel \mathbf{\Theta}'_{11} \\ \mathbf{\Theta}'_{12} \parallel \mathbf{\Theta}'_{13} \parallel \mathbf{\Theta}'_{14} \parallel \mathbf{\Theta}'_{15} \end{pmatrix} = \begin{pmatrix} \mathbf{\Delta}_0 \parallel \mathbf{\Delta}_{13} \parallel \mathbf{\Delta}_{10} \parallel \mathbf{\Delta}_7 \\ \mathbf{\Delta}_4 \parallel \mathbf{\Delta}_1 \parallel \mathbf{\Delta}_{14} \parallel \mathbf{\Delta}_{11} \\ \mathbf{\Delta}_8 \parallel \mathbf{\Delta}_5 \parallel \mathbf{\Delta}_2 \parallel \mathbf{\Delta}_{15} \\ \mathbf{\Delta}_{12} \parallel \mathbf{\Delta}_9 \parallel \mathbf{\Delta}_6 \parallel \mathbf{\Delta}_3 \end{pmatrix}$$

E Proof of Corollary 1

Proof. We first present some observations on the pre-computed matrices $\mathbf{M}_{(\alpha,\alpha,\beta)}$ for any $\alpha, \beta \in \mathbf{F}_{2^8}$.

- The matrices $\mathbf{M}_{(\alpha,\alpha,0)}$ for any $\alpha \in \mathbf{F}^*_{2^8}$ are always equal to $\begin{pmatrix} \frac{1}{512} & -\frac{1}{512} \\ -\frac{1}{512} & \frac{1}{512} \end{pmatrix}$;
- The matrices $\mathbf{M}_{(0,0,\beta)}$ for any $\beta \in \mathbf{F}^*_{2^8}$ always have the form of $\begin{pmatrix} val & val \\ -val & -val \end{pmatrix}$, where val is a rational number.
- The matrix $\mathbf{M}_{(0,0,0)}$ is equal to $\begin{pmatrix} \frac{257}{512} & \frac{255}{512} \\ \frac{255}{512} & \frac{257}{512} \end{pmatrix}$.

For any 32-bit masks $\mathbf{A} = (\mathbf{a}_0 \parallel \mathbf{a}_1 \parallel \mathbf{a}_2 \parallel \mathbf{a}_3)$ and $\mathbf{B} = (\mathbf{b}_0 \parallel \mathbf{b}_1 \parallel \mathbf{b}_2 \parallel \mathbf{b}_3)$, we have $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B}) = \boldsymbol{l}_2\mathbf{M}_{(\mathbf{a}_3,\mathbf{a}_3,\mathbf{b}_3)}\mathbf{M}_{(\mathbf{a}_2,\mathbf{a}_2,\mathbf{b}_2)}\mathbf{M}_{(\mathbf{a}_1,\mathbf{a}_1,\mathbf{b}_1)}\mathbf{M}_{(\mathbf{a}_0,\mathbf{a}_0,\mathbf{b}_0)}\boldsymbol{e}_0 \neq 0$. According to the above observations,

(1) If $\mathbf{a}_3$=`0x00`, we must have $\mathbf{b}_3$=`0x00`. Otherwise $\boldsymbol{l}_2\mathbf{M}_{(0,0,\mathbf{b}_3)} = (0, 0)$, and thus $\mathrm{Cor}_3(\mathbf{A}; \mathbf{A}, \mathbf{B}) = 0$. Similarly, we can prove that, if $\mathbf{b}_3$=`0x00`, we must have $\mathbf{a}_3$=`0x00`.

(2) If $\mathbf{a}_3 = \mathbf{a}_2 =$0x00, from the above we have $\mathbf{b}_3=$0x00, and $\boldsymbol{l}_2\mathbf{M}_{(\mathbf{a}_3,\mathbf{a}_3,\mathbf{b}_3)} = \boldsymbol{l}_2$. Besides, we must have $\mathbf{b}_2=$0x00. Otherwise $\boldsymbol{l}_2\mathbf{M}_{(\mathbf{a}_3,\mathbf{a}_3,\mathbf{b}_3)}\mathbf{M}_{(0,0,\mathbf{b}_2)} = \boldsymbol{l}_2\mathbf{M}_{(0,0,\mathbf{b}_2)} = (0,0)$, and thus $\mathrm{Cor}_3(\mathbf{A};\mathbf{A},\mathbf{B}) = 0$. Similarly, we can prove that, if $\mathbf{b}_3 = \mathbf{b}_2 =$0x00, we must have $\mathbf{a}_3 = \mathbf{a}_2 =$0x00.

(3) If $\mathbf{a}_3 = \mathbf{a}_2 = \mathbf{a}_1 =$0x00, we have $\mathbf{b}_3 = \mathbf{b}_2 =$0x00, and $\boldsymbol{l}_2\mathbf{M}_{(\mathbf{a}_3,\mathbf{a}_3,\mathbf{b}_3)}\mathbf{M}_{(\mathbf{a}_2,\mathbf{a}_2,\mathbf{b}_2)} = \boldsymbol{l}_2$. Besides, we must have $\mathbf{b}_1=$0x00. Otherwise $\boldsymbol{l}_2\mathbf{M}_{(\mathbf{a}_3,\mathbf{a}_3,\mathbf{b}_3)}\mathbf{M}_{(\mathbf{a}_2,\mathbf{a}_2,\mathbf{b}_2)}\mathbf{M}_{(0,0,\mathbf{b}_1)} = \boldsymbol{l}_2\mathbf{M}_{(0,0,\mathbf{b}_1)} = (0,0)$, and thus $\mathrm{Cor}_3(\mathbf{A};\mathbf{A},\mathbf{B}) = 0$. Similarly, we can prove that, if $\mathbf{b}_3 = \mathbf{b}_2 = \mathbf{b}_1 =$0x00, we must have $\mathbf{a}_3 = \mathbf{a}_2 = \mathbf{a}_1 =$0x00.

(4) If $\mathbf{a}_3 = \mathbf{a}_2 = \mathbf{a}_1 = \mathbf{a}_0 =$0x00, we have $\mathbf{b}_3 = \mathbf{b}_2 = \mathbf{b}_1 =$0x00, and $\boldsymbol{l}_2\mathbf{M}_{(\mathbf{a}_3,\mathbf{a}_3,\mathbf{b}_3)}\mathbf{M}_{(\mathbf{a}_2,\mathbf{a}_2,\mathbf{b}_2)}\mathbf{M}_{(\mathbf{a}_1,\mathbf{a}_1,\mathbf{b}_1)} = \boldsymbol{l}_2$. Besides, we must have $\mathbf{b}_0 \neq$0x00. Otherwise $\mathrm{Cor}_3(\mathbf{A};\mathbf{A},\mathbf{B}) = 0$. Similarly, we can prove that, if $\mathbf{b}_3 = \mathbf{b}_2 = \mathbf{b}_1 = \mathbf{b}_0 =$0x00, we must have $\mathbf{a}_3 = \mathbf{a}_2 = \mathbf{a}_1\mathbf{a}_0 =$0x00.

(5) If $\mathbf{A} = \mathbf{B} =$0x00000000, $\mathrm{Cor}_3(\mathbf{A};\mathbf{A},\mathbf{B}) = \boldsymbol{l}_2\mathbf{M}_{(0,0,0)}\mathbf{M}_{(0,0,0)}\mathbf{M}_{(0,0,0)}\mathbf{M}_{(0,0,0)}\boldsymbol{e}_0 = 1$.

Above all, we complete the proof. □

F All 255 choices for $\Lambda \in \mathcal{S}_{id}$ such that $\Lambda' \in \overline{\mathcal{S}}_{id}$

Table 3: All 255 choices of $\mathbf{\Lambda}_{id}$ for $\mathbf{\Lambda} \in \mathcal{S}_{id}$ such that $\mathbf{\Lambda}' \in \overline{\mathcal{S}}_{id}$

0x014893ff,	0x02d9b501,	0x039126fe,	0x04b36b03,	0x05fbf8fc,	0x066ade02,	0x07224dfd,
0x082e45f8,	0x0966d607,	0x0af7f0f9,	0x0bbf6306,	0x0c9d2efb,	0x0dd5bd04,	0x0e449bfa,
0x0f0c0805,	0x1014190f,	0x115c8af0,	0x12cdac0e,	0x13853ff1,	0x14a7720c,	0x15efe1f3,
0x167ec70d,	0x173654f2,	0x183a5cf7,	0x1972cf08,	0x1ae3e9f6,	0x1bab7a09,	0x1c8937f4,
0x1dc1a40b,	0x1e5082f5,	0x1f18110a,	0x2029321f,	0x2161a1e0,	0x22f0871e,	0x23b814e1,
0x249a591c,	0x25d2cae3,	0x2643ec1d,	0x270b7fe2,	0x280777e7,	0x294fe418,	0x2adec2e6,
0x2b965119,	0x2cb41ce4,	0x2dfc8f1b,	0x2e6da9e5,	0x2f253a1a,	0x303d2b10,	0x3175b8ef,
0x32e49e11,	0x33ac0dee,	0x348e4013,	0x35c6d3ec,	0x3657f512,	0x371f66ed,	0x38136ee8,
0x395bfd17,	0x3acadbe9,	0x3b824816,	0x3ca005eb,	0x3de89614,	0x3e79b0ea,	0x3f312315,
0x4052643f,	0x411af7c0,	0x428bd13e,	0x43c342c1,	0x44e10f3c,	0x45a99cc3,	0x4638ba3d,
0x477029c2,	0x487c21c7,	0x4934b238,	0x4aa594c6,	0x4bed0739,	0x4ccf4ac4,	0x4d87d93b,
0x4e16ffc5,	0x4f5e6c3a,	0x50467d30,	0x510eeecf,	0x529fc831,	0x53d75bce,	0x54f51633,
0x55bd85cc,	0x562ca332,	0x576430cd,	0x586838c8,	0x5920ab37,	0x5ab18dc9,	0x5bf91e36,
0x5cdb53cb,	0x5d93c034,	0x5e02e6ca,	0x5f4a7535,	0x607b5620,	0x6133c5df,	0x62a2e321,
0x63ea70de,	0x64c83d23,	0x6580aedc,	0x66118822,	0x67591bdd,	0x685513d8,	0x691d8027,
0x6a8ca6d9,	0x6bc43526,	0x6ce678db,	0x6daeeb24,	0x6e3fcdda,	0x6f775e25,	0x706f4f2f,
0x7127dcd0,	0x72b6fa2e,	0x73fe69d1,	0x74dc242c,	0x7594b7d3,	0x7605912d,	0x774d02d2,
0x78410ad7,	0x79099928,	0x7a98bfd6,	0x7bd02c29,	0x7cf261d4,	0x7dbaf22b,	0x7e2bd4d5,
0x7f63472a,	0x80a4c97f,	0x81ec5a80,	0x827d7c7e,	0x8335ef81,	0x8417a27c,	0x855f3183,
0x86ce177d,	0x87868482,	0x888a8c87,	0x89c21f78,	0x8a533986,	0x8b1baa79,	0x8c39e784,
0x8d71747b,	0x8ee05285,	0x8fa8c17a,	0x90b0d070,	0x91f8438f,	0x92696571,	0x9321f68e,
0x9403bb73,	0x954b288c,	0x96da0e72,	0x97929d8d,	0x989e9588,	0x99d60677,	0x9a472089,
0x9b0fb376,	0x9c2dfe8b,	0x9d656d74,	0x9ef44b8a,	0x9fbcd875,	0xa08dfb60,	0xa1c5689f,
0xa2544e61,	0xa31cdd9e,	0xa43e9063,	0xa576039c,	0xa6e72562,	0xa7afb69d,	0xa8a3be98,
0xa9eb2d67,	0xaa7a0b99,	0xab329866,	0xac10d59b,	0xad584664,	0xaec9609a,	0xaf81f365,
0xb099e26f,	0xb1d17190,	0xb240576e,	0xb308c491,	0xb42a896c,	0xb5621a93,	0xb6f33c6d,
0xb7bbaf92,	0xb8b7a797,	0xb9ff3468,	0xba6e1296,	0xbb268169,	0xbc04cc94,	0xbd4c5f6b,
0xbedd7995,	0xbf95ea6a,	0xc0f6ad40,	0xc1be3ebf,	0xc22f1841,	0xc3678bbe,	0xc445c643,
0xc50d55bc,	0xc69c7342,	0xc7d4e0bd,	0xc8d8e8b8,	0xc9907b47,	0xca015db9,	0xcb49ce46,
0xcc6b83bb,	0xcd231044,	0xceb236ba,	0xcffaa545,	0xd0e2b44f,	0xd1aa27b0,	0xd23b014e,
0xd37392b1,	0xd451df4c,	0xd5194cb3,	0xd6886a4d,	0xd7c0f9b2,	0xd8ccf1b7,	0xd9846248,
0xda1544b6,	0xdb5dd749,	0xdc7f9ab4,	0xdd37094b,	0xdea62fb5,	0xdfeebc4a,	0xe0df9f5f,
0xe1970ca0,	0xe2062a5e,	0xe34eb9a1,	0xe46cf45c,	0xe52467a3,	0xe6b5415d,	0xe7fdd2a2,
0xe8f1daa7,	0xe9b94958,	0xea286fa6,	0xeb60fc59,	0xec42b1a4,	0xed0a225b,	0xee9b04a5,
0xefd3975a,	0xf0cb8650,	0xf18315af,	0xf2123351,	0xf35aa0ae,	0xf478ed53,	0xf5307eac,
0xf6a15852,	0xf7e9cbad,	0xf8e5c3a8,	0xf9ad5057,	0xfa3c76a9,	0xfb74e556,	0xfc56a8ab,
0xfd1e3b54,	0xfe8f1daa,	0xffc78e55				