Received December 6, 2021, accepted December 27, 2021, date of publication January 7, 2022, date of current version May 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141434

Key-Dependent Feedback Configuration Matrix of Primitive σ –LFSR and Resistance to Some Known Plaintext Attacks

SUBRATA NANDI[1], SRINIVASAN KRISHNASWAMY[2], BEHROUZ ZOLFAGHARI[1], AND PINAKI MITRA[1]
[1]CSE Department, Indian Institute of Technology Guwahati, Guwahati 781039, India
[2]EEE Department, Indian Institute of Technology Guwahati, Guwahati 781039, India
Corresponding author: Subrata Nandi (subrata.nandi@iitg.ac.in)

ABSTRACT In this paper, we propose and evaluate a method for generating key-dependent feedback configurations (KDFC) for σ-LFSRs. σ-LFSRs with such configurations can be applied to any stream cipher that uses a word-based LFSR. Here, a configuration generation algorithm uses the secret key(K) and the Initialization Vector (IV) to generate a new feedback configuration after the initialization round. It replaces the older known feedback configuration. The keystream is generated from this new feedback configuration and the FSM part. We have mathematically analysed the feedback configurations generated by this method. As a test case, we have applied this method on SNOW 2.0 and have studied its impact on resistance to algebraic attacks. Besides, as a consequence of resisting algebraic attacks, SNOW 2.0 can also withstand some other attacks like Distinguishing Attack, Fast Correlation Attack, Guess and Determining Attack and Cache Timing Attack. Further, we have also tested the generated keystream for randomness and have briefly described its implementation and the challenges involved in the same.

INDEX TERMS Stream Cipher σ-LFSR key-dependent feedback configuration primitive polynomial algebraic attack.

I. INTRODUCTION

Stream ciphers are used in a variety of applications [1], [2]. LFSRs (Linear Feedback Shift Register) are widely used as building blocks in stream ciphers([3], [4]) because of their simple construction, good pseudorandomness([5]) and easy implementation.

Word-based LFSRs were introduced to efficiently use the structure of modern word-based processors ([6]–[13]). Such LFSRs are used in a variety of stream ciphers, most notably in the SNOW series of stream ciphers. A σ-LFSR is a word-based LFSR configuration that was introduced in [14]. An important property of this configuration is that multiple feedback functions are corresponding to a given characteristic polynomial of the state transition matrix([15]). The number of such configurations was conjectured in ([14]). This conjecture was constructively proved in ([16]).

The knowledge of the feedback function plays a critical role in most attacks on LSFR based stream ciphers. These include algebraic attacks, correlation attacks, distinguishing attacks, guess and determining attack and Cache timing attack ([17]–[21]). Therefore, hiding the feedback function of the LFSR could potentially increase the security of such schemes. One way of doing this is by using dynamic feedback control. This approach is used in stream ciphers such as K2 ([22]) and A5. This converts the deterministic linear recurrence of some registers into a probabilistic recurrence. However, key recovery attacks on K2 and A5 have been reported in the literature ([23], [24]).

In this paper, we try to increase the security of LFSR based word-oriented stream ciphers by making the feedback function dependent on the secret key. The resulting configuration is called the σ-KDFC (Key Dependent Feedback Configuration). The proposed method for obtaining the feedback function from the secret key utilizes the algorithm given in [15], [16]. The feedback gains thus obtained are highly non-linear functions of the secret key. Further, the number of iterations in this algorithm can be adjusted depending on the available computing power. As an example, we study the interconnection of the σ-KDFC with the finite state machine (FSM) of

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

SNOW-2. We use empirical tests to verify the randomness of the keystream generated by this scheme. Further, we analyse the scheme for security against various kinds of attacks.

TABLE 1. Symbol and their meaning.

Symbol	Meaning
$\mathbb{F}_{2^n}$	Finite field of cardinality 2^n
$\mathbb{F}_2^n$	n-dimensional vector space over $\mathbb{F}_2$
$M[i,:]$	The i^{th} row of a matrix M
$M[:,j]$	j^{th} column of a matrix M
$M[i,j]$	(i,j)-th entry of the matrix M
$\mu(M[i,j])$	The minor of $M[i,j]$
$\oplus$	XOR
$+$	Addition or XOR
$M^{n\times n}$	Matrix M with n rows and n columns
e_1^n	Vector with $n-$th coordinate is 1 and rest are 0.
M^{-1}	Inverse of a matrix M.

A. SYMBOL TABLE

The rest of this paper is organized as follows. Section II-A introduces LFSRs, σ-LFSRs and some related concepts. Section III examines σ-KDFC and its time complexity. Section IV presents the mathematical analysis on the algebraic degree of the parameters of the feedback function generated by σ-KDFC. Section V discusses the interconnection of σ-KDFC with the FSM of SNOW and its security against various cryptographic attacks. Section VI concludes the paper.

B. CONTRIBUTION

The contribution of this article is as follows:

- We have proposed a method of making of making the feedback configuration of a word based shift register dependent on the key of a stream cipher. Most Known plaintext attacks (KPAs) on LFSR based stream ciphers, particularly algebraic ones, make use of the feedback equation. Therefore, making the feedback equation dependent on the secret key protects the cipher against such attacks.
- We have also calculated the algebraic degree of the entries of the resulting feedback matrices.
- As a test case, we have analysed the security of SNOW 2.0 after replacing the LFSR in that cipher with the proposed configuration.
- Finally, we have tested the pseudorandomness of KDFC-SNOW with 16 NIST randomness tests.

II. PRELIMINARY CONCEPTS

A. LFSRs AND σ-LFSRs

An LFSR is an electronic circuit that implements a Linear Recurring Relation (LRR) of the form $x_{n+b} = a_{b-1}x_{n+b-1} + \cdots + a_0x_n$. It consists of a shift register with b flip-flops and a linear feedback function which is typically implemented using XOR gates. The integer b is called the length of the LFSR. The characteristic polynomial of an LFSR is a monic polynomial with the same coefficients as the LRR implemented by it. For example, the characteristic polynomial of an LFSR which implements the LRR given above is $x^b + a_{b-1}x^{b-1} + \cdots + a_0$. The period of the sequence generated by an LFSR of length b is at most $2^b - 1$. Further, an LFSR generates a maximum-period sequence if its characteristic polynomial is primitive [25]. The state vector of an LFSR at any time instant is a vector whose entries are the outputs of the delay blocks at the time instant i.e $\mathbf{x_n} = [x_n, x_{n+1}, \ldots, x_{n+b-1}]$. Two consecutive state vectors are related by the equation $\mathbf{x_{n+1}} = \mathbf{x_n}P_f$ where P_f is the companion matrix of the characteristic polynomial.

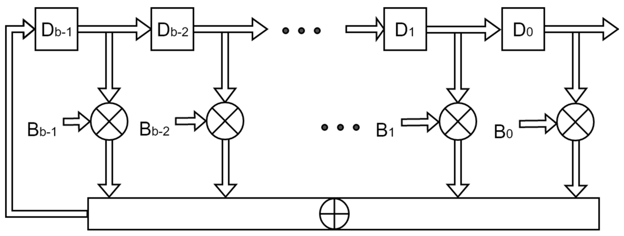


FIGURE 1. Block diagram of σ-LFSR.

$$P_f = \begin{bmatrix} 0 & 0 & \ldots. & 0 & a_0 \\ 1 & 0 & \ldots. & 0 & a_1 \\ 0 & 1 & \ldots. & 0 & a_2 \\ \vdots & \vdots & \ldots. & \vdots & \vdots \\ 0 & 0 & \ldots. & 1 & a_{b-1} \end{bmatrix} \tag{1}$$

In order to efficiently work with word based processors various word based LFSR designs have been proposed [6]–[8], [11], [13], [22], [26]. These designs use multi input multi output delay blocks. One such design is the σ-LFSR shown in Figure 1. Here, the feedback gains are matrices and the implemented linear recurring relation is of the form

$$\mathbf{x_{n+b}} = B_{n+b-1}\mathbf{x_{n+b-1}} + B_{n+b-2}\mathbf{x_{n+b-2}} + \cdots + B_0\mathbf{x_n} \tag{2}$$

where each $\mathbf{x_i} \in \mathbb{F}_2^m$ and $B_i \in \mathbb{F}_2^{m\times m}$. Here, each delay block has m-inputs and m-outputs and the σ-LFSR generates a sequence of vectors in $\mathbb{F}_2^m$

The matrices $B_0, B_1, \cdots, B_{b-1}$ are referred to as the gain matrices of the σ-LFSR and the following matrix is defined as its configuration matrix.

$$C = \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & 0 & \cdots & I \\ B_0 & B_1 & B_2 & \cdots & B_{b-1} \end{bmatrix} \in \mathbb{F}_2^{mb\times mb} \tag{3}$$

where $0, I \in \mathbb{F}_2^{m\times m}$ are the all-zero and identity matrices respectively. We shall refer to the structure of this matrix as the M-companion structure. The characteristic polynomial of this configuration matrix is known as the characteristic polynomial of the σ-LFSR.

If $\mathbf{x_n}$ is the output of the σ-LFSR at the n-th time instant, then its state vector at that time instant is defined as

$\hat{\mathbf{x}_n} = [\mathbf{x_n}, \mathbf{x_{n+1}}, \ldots, \mathbf{x_{n+b-1}}]$. This vector is got by stacking the outputs of all the delay blocks at the n-th time instant. Two consecutive state vectors are related by the following equation:

$$\hat{\mathbf{x}_{n+1}} = \hat{\mathbf{x}_n} C^T \tag{4}$$

In the case of σ-LFSRs, there are many possible feedback configurations having the same characteristic polynomial. For a given primitive polynomial, the number of such configurations was conjectured in [14] to be the following,

$$N_P = \frac{|GL(m, \mathbb{F}_2)|}{2^m - 1} \times \frac{\phi(2^{mb} - 1)}{mb} \times 2^{m(m-1)(b-1)} \tag{5}$$

where 5, $GL(m, \mathbb{F}_2)$ is the general linear group of non-singular matrices $\in \mathbb{F}_2^{m \times m}$, and ϕ represents Euler's totient. This conjecture has been proved in [27]. Moreover, this inductive proof is constructive and gives an algorithm for calculating such feedback functions.

In the following section, we shall use the algorithm given in [27] to develop a key dependent feedback configuration for the σ-LFSR.

III. σ-KDFC

Stream ciphers, like the SNOW series of ciphers, use word based LFSRs along with an FSM module. The feedback configuration of the LFSR in such schemes is publicly known. This feedback relation is used in most attacks on such schemes [17], [19], [21], [28]. Therefore, the security of such schemes could potentially increase if the feedback function is made key dependent.

Before proceeding to our construction of a key dependent feedback configuration, we briefly describe the algorithm given in [27] which generates feedback configurations for σ-LFSRs with a given characteristic polynomial. Given a primitive polynomial $p_{mb}(x)$ having degree mb, the algorithm for calculating a feedback configuration for a σ-LFSR with b m-input m-output delay blocks is as follows:

Algorithm 1 Configuration Matrix Generation Algorithm

```
1:  Initialize $Y$ with a full rank matrix in $\mathbb{F}_2^{m\times m}$.
2:  Choose $mb-m-1$ primitive polynomials $p_m(x), p_{m+1}(x), \ldots, p_{mb-1}(x)$ having degrees $m.m+1, \ldots, mb-1$ respectively. (Note that the polynomial $p_{mb}(x)$ is given). Let $A_m, A_{m+1}, \ldots, A_{mb}$ be the companion matrices of $p_m(x), p_{m+1}(x), \ldots, p_{mb}(x)$ respectively.
3:  for $i = 1$ to $mb - m$ do
4:      Let $\ell$ be the unique integer less or equal to $m$ which is equivalent to $i$ mod $m$. Find a polynomial $f(x)$ such that $Y[\ell \bmod m, :] \times A_{m+i-1} = (0, 0, \ldots, 1) \in \mathbb{F}_2^{m+i-1}$ and update $Y$ as follows
            $Y \leftarrow Y * A_{m+i-1}$
5:      if $t \neq \ell$ mod $m$ then
            $Y[t, :] \leftarrow (Y[t, :], d_t)$ where $d_t$ is randomly sampled element of $\mathbb{F}_2$.(In this step all the rows of $Y$, except the one which is equivalent to $i$ mod $m$, are appended with random boolean numbers and their lengths are increased by 1.)
6:      end if
7:      if $t == \ell$ then
8:          $Y[t, :] \leftarrow (0, 0, \ldots, 1) \in \mathbb{F}_2^{m+i}$
9:      end if
10: end for(At the end of each iteration, an extra column is added to $Y$ till $Y \in \mathbb{F}_2^{m\times mb}$).
11: Construct the following matrix $Q$.
```

$$Q \leftarrow \begin{bmatrix} Y[0:,] \\ Y[1:,] \\ \vdots \\ Y[m-1:,] \\ Y[0:,]\times P_{mb} \\ Y[1:,]\times P_{mb} \\ \vdots \\ Y[m-1:,]\times P_{mb} \\ \vdots \\ Y[0:,]\times (P_{mb})^{b-1} \\ Y[1:,]\times (P_{mb})^{b-1} \\ \vdots \\ Y[m-1:,]\times (P_{mb})^{b-1} \end{bmatrix} \in \mathbb{F}_2^{mb\times mb}$$

```
12: $C \leftarrow Q \times P_{mb} \times Q^{-1}$
13: return $C$ as Configuration Matrix of $p_{mb}(x)$.
```

The matrix C generated in the above algorithm is the configuration matrix of a σ-LFSR with characteristic polynomial $p_{mb}(x)$. As can be seen from Equation 3, the last m rows of this matrix contain the feedback gain matrices. Each set of choices for the d_ts and the initial full rank matrix result in a different feedback configuration.

In Step 3a, the coefficients of the polynomial $f(x)$ can be calculated by solving the linear equation $y \times K_t = (0, 0, \ldots, 1) \in \mathbb{F}_2^{m+i-1}$ for y, where K_i is given by

$$K_i = \begin{bmatrix} Y[\ell, :] \\ Y[\ell, :]A_{m+i-1} \\ \vdots \\ Y[\ell, :]A_{m+i-1}^{m+i-2} \end{bmatrix}$$

In other words $f(x) = y(1) + y(2)x + \cdots + y(m+i-1)x^{m+i-2}$, where $y(j)$ is the j-th entry of the vector y. Note that in every iteration of Step 3 in Algorithm 1, $m-1$ random numbers are appended to the rows of the matrix Y. In the proposed scheme, some of these numbers are derived from the secret key. As a consequence, the derived feedback configuration is dependent on the secret key. We now proceed to look at this configuration in detail.

Example 1: For a better understanding, we demonstrate the generation of an invertable matrix Q for a $\sigma-$LFSR with 2 4-input 4-output delay blocks, i.e. m = 4, b = 2. The characteristic polynomial of the LFSR is assumed to be

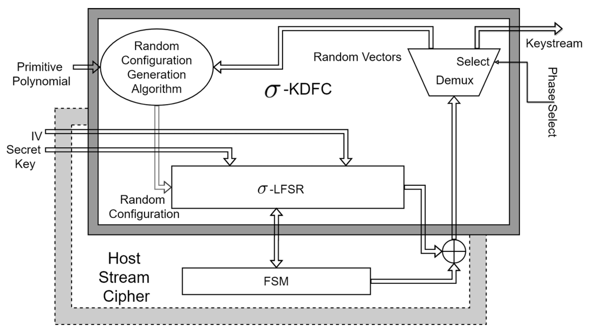


FIGURE 2. The Schematic of σ-KDFC.

$x^8 + x^4 + x^3 + x^2 + 1$ which is a primitive polynomial of degree 8.

Ist Iteration: In this iteration the matrix M is the companion matrix of the primitive polynomial $x^4 + x + 1$.

$$\begin{pmatrix} 1&1&0&1\\0&1&0&0\\1&0&1&1\\0&0&0&1 \end{pmatrix}_{4\times4} \xrightarrow{xM^i} \begin{pmatrix} 0&0&0&1\\1&1&1&1\\1&0&0&1\\1&0&1&1 \end{pmatrix} \mapsto \begin{pmatrix} 0&0&0&0&1\\1&1&1&1&1\\1&0&0&1&1\\1&0&1&1&0 \end{pmatrix}_{4\times5}$$

2nd Iteration: In this iteration the matrix M is the companion matrix of the primitive polynomial $x^5 + x^2 + 1$.

$$\begin{pmatrix} 0&0&0&0&1\\1&1&1&1&1\\1&0&0&1&1\\1&0&1&1&0 \end{pmatrix}_{4\times5} \xrightarrow{xM^i} \begin{pmatrix} 0&0&0&1&1&1\\0&0&0&0&0&1\\1&0&1&0&0&1\\1&1&0&1&0&0 \end{pmatrix}_{4\times6}$$

3rd Iteration: In this iteration the matrix M is the companion matrix of the primitive polynomial $x^6 + x + 1$.

$$\begin{pmatrix} 0&0&0&1&1&1\\0&0&0&0&0&1\\1&0&1&0&0&1\\1&1&0&1&0&0 \end{pmatrix} \xrightarrow{xM^i} \begin{pmatrix} 1&0&0&0&1&0&1\\0&0&1&1&0&1&0\\0&0&0&0&0&0&1\\0&0&0&1&1&0&1 \end{pmatrix}$$

4th Iteration: In this iteration the matrix M is the companion matrix of the primitive polynomial $x^7 + x + 1$.

$$\begin{pmatrix} 1&0&0&0&1&0&1\\0&0&1&1&0&1&0\\0&0&0&0&0&0&1\\0&0&0&1&1&0&1 \end{pmatrix} \xrightarrow{xM^i} \begin{pmatrix} 0&0&1&0&0&1&1&1\\0&0&0&0&0&1&0&1\\1&0&0&1&1&1&0&1\\0&0&0&0&0&0&0&1 \end{pmatrix}$$

(In the above iterations, the colour red indicates that the number has been randomly sampled). At the end of the above iterations we get the following matrix Y

$$Y = \begin{pmatrix} 0&0&1&0&0&1&1&1\\0&0&0&0&0&1&0&1\\1&0&0&1&1&1&0&1\\0&0&0&0&0&0&0&1 \end{pmatrix}_{4\times4}$$

The corresponding matrix Q is as follows.

$$Q = \begin{pmatrix} 0&0&1&0&0&1&1&1\\0&0&0&0&0&1&0&1\\1&0&0&1&1&1&0&1\\0&0&0&0&0&0&0&1\\0&1&0&0&1&1&1&1\\0&0&0&0&1&0&1&0\\0&0&1&1&1&0&1&1\\0&0&0&0&0&0&1&0 \end{pmatrix}_{8\times8}$$

The following matrix P is the companion matrix of the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$.

$$P = \begin{pmatrix} 0&0&0&0&0&0&0&1\\1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&1\\0&0&1&0&0&0&0&1\\0&0&0&1&0&0&0&1\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0 \end{pmatrix}_{8\times8}$$

This results in the following matrix C

$$C = Q * P * Q^{-1} = \left(\begin{array}{cccc|cccc} 0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\\ \hline 0&0&1&0&0&0&0&1\\1&0&0&1&0&1&1&1\\0&0&0&1&1&0&1&1\\0&1&0&1&0&0&0&0 \end{array}\right)_{8\times8}$$

$$C = \left(\begin{array}{c|c} 0 & I \\ \hline B_0 & B_1 \end{array}\right)$$

In the above example, B_0 and B_1 are the two gain matrices that we have computed using Algorithm 1.

In order to create a keystream generator from the proposed σ-LFSR configuration, it can be connected to a Finite state machine which introduces non-linearity. Figure 2 shows the schematic of the proposed scheme along with its interconnection with an FSM. The scheme has an initialization phase wherein the feedback configuration of the σ-LFSR is calculated by running Algorithm 1. In order to reduce the time taken for initialization, Algorithm 1 is run offline (at some server) till k iterations of step 3 and the resulting matrix Y is made public. The number k can be chosen depending on the computational capacity of the machine that hosts the σ-LFSR. The feedback configuration is calculated by running the remaining part of the algorithm in the initialization phase. In this phase there is no keystream generated at the output. The following subsection explains the initialization phase in detail.

A. THE INITIALIZATION PHASE

During the initialization phase, the σ-LFSR has a publicly known feedback configuration. Further, the pre-calculated matrix $Y \in \mathbb{F}_2^{m\times(m+k)}$, and the primitive polynomials $p_{m+k+1}(x), p_{m+k+2}(x), \ldots, p_{mb}(x)$ are also publicly known. The initial state of the σ-LFSR is derived from the secret key and the IV. (as is normally done in word based stream ciphers like SNOW). The σ-LFSR is run along with the FSM for $mb-m-k$ clock cycles. This generates $mb-m-k$ vectors in $\mathbb{F}_2^m$. This corresponds to the $mb-m-k$ remaining iterations of Step 3 in Algorithm 1.

The remaining part of Algorithm 1 is now run. In each iteration of Step 3, the boolean numbers appended to the rows of the matrix Y in Step 3(b) are the entries of the corresponding vector. More precisely, in the i-th iteration of Step 3 that is run on the keystream generator, for $t \neq i+k \mod m$, the t-th row of Y is appended with the t-th entry of the i-th vector that was generated.

The feedback gains of the σ-LFSR are now set according to the configuration matrix that is generated by Algorithm 1.

Once the feedback gains are set the σ-LFSR is run along with the FSM. The first b vectors are discarded and the keystream starts from the $b+1$-th vector. The reason for doing this is that the initial state of the σ-LFSR with the new configuration is generated by the publicly known feedback configuration which is used in the initialization process.

The algorithm for generating the configuration matrix can be applied for all values of m and b. Therefore, the above described KDFC scheme can be used along with any existing word based stream cipher irrespective of the size and number of delay blocks.

1) TIME COMPLEXITY OF THE INITIALIZATION PHASE

Step 3(a) in Algorithm 1 involves solving a system of linear equations in less than mb variables. This can be done with a time complexity of $\mathcal{O}((mb)^3)$ using Gaussian elimination. The time complexity of Step 3(b) is linear in m while that of Step 3(c) is constant. Further, Step 3 has $mb-m-k$ iterations. Therefore, if k is chosen such that $mb-m-k$ is $\mathcal{O}(1)$, then the overall time complexity of Step 3 is $\mathcal{O}((mb)^3)$. In Step 5, the matrix C can be calculated by solving the linear system of equations $CQ = QP_{mb}$ for C. This can be done in $\mathcal{O}((mb)^4)$ using Gaussian elimination. Step 4 has a time complexity of $\mathcal{O}((mb)^3)$. Thus, the time complexity of the initialization phase is $\mathcal{O}((mb)^4)$.

IV. ALGEBRAIC ANALYSIS OF σ-KDFC

The entries of the feedback matrices, $B_0, B_1, \ldots, B_{b-1}$, calculated by the procedure given in the previous section are functions of the matrix Y generated in Step 3 of Algorithm 1. The entries of Y are in turn non-linear functions of the initial state of the σ-LFSR.

Note that the last row of Y is always e_1^n. Let the first $m-1$ rows of Y be $v_1, v_2, \ldots, v_{m-1}$. Let $\mathcal{U}$ be the set of variables that denote the entries in these rows. Therefore,

$$B_k(i,j) = f_{k(i,j)}(\mathcal{U}) \text{ for } 0 \le k \le b-1 \text{ and } 1 \le i,j \le m \tag{6}$$

where $f_{k(i,j)}$s are polynomial functions.

The algebraic degree of the configuration matrix, denoted by Θ, is defined as follows

$$\Theta\left(C_{\mathcal{S}}\right) = \max_{k,i,j}\left(|f_{k(i,j)}(\mathcal{U})|\right) \tag{7}$$

Θ can be considered as a measure of the algebraic resistance of σ-KDFC. We now proceed to find a lower bound for Θ.

The matrix Q generated in Step 4 of Algorithm 1 is given as follows

$$Q = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{m-1} \\ e_1^n \\ v_1 P_{mb} \\ \vdots \\ v_{m-1}P_{mb} \\ e_1^n P_{mb} \\ \vdots \\ v_1 P_{mb}^{b-1} \\ \vdots \\ v_{m-1}P_{mb}^{b-1} \\ e_1 P_{mb}^{b-1} \end{bmatrix} \tag{8}$$

where P_{mb} is the companion matrix of the publicly known primitive characteristic polynomial of the σ-LFSR. The configuration matrix C is generated by the formula $C = Q \times P_{mb} \times Q^{-1}$. Since Q is an invertable boolean matrix, the determinant of Q is always 1. Therefore, $Q^{-1} = Q^{(a)}$ where $Q^{(a)}$ is the adjugate of Q. Moreover, since the elements of Q belong to $\mathbb{F}_2$, the co-factors are equal to the minors of Q. The rows of Q can be permuted to get the following matrix Q_P

$$Q^P = \begin{bmatrix} e_1^n \\ e_1^n P_{mb} \\ \vdots \\ e_1 P_{mb}^{b-1} \\ v_1 \\ v_1 P_{mb} \\ \vdots \\ v_1 P_{mb}^{b-1} \\ \vdots \\ v_{m-1} \\ v_{m-1}P_{mb} \\ \vdots \\ v_{m-1}P_{mb}^{b-1} \end{bmatrix} \tag{9}$$

The matrix Q_P can be decomposed as follows into four submatrices Q_1, Q_2, Q_3 and Q_4:

$$Q_P = \left[\begin{array}{c|c} Q_1 & Q_2 \\ \hline Q_3 & Q_4 \end{array}\right] \tag{10}$$

where $Q_1 \in \mathbb{F}_2^{b\times(mb-b)}$ is the all zero matrix and the matrices $Q_2 \in \mathbb{F}_2^{b\times b}$, $Q_3 \in \mathbb{F}_2^{mb-b\times mb-b}$ and $Q_4 \in \mathbb{F}_2^{mb-b\times b}$ are as follows

$$Q_2 = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & * \\ \vdots & \vdots & & \\ 1 & \cdots & * & * \end{pmatrix} \tag{11}$$

$$Q_3 = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,mb-b} \\ v_{1,2} & v_{1,3} & \cdots & v_{1,mb-b+1} \\ \vdots & \vdots & \cdots & \\ v_{1,b} & v_{1,b+1} & \cdots & v_{1,mb} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,mb-b} \\ v_{2,2} & v_{2,3} & \cdots & v_{2,mb-b+1} \\ \vdots & \vdots & \cdots & \vdots \\ v_{2,b} & v_{2,b+1} & \cdots & v_{2,mb} \\ \vdots & \vdots & \cdots & \vdots \\ v_{m-1,1} & v_{m-1,2} & \cdots & v_{m-1,mb-b} \\ v_{m-1,2} & v_{m-1,3} & \cdots & v_{m-1,mb-b+1} \\ \vdots & \vdots & \cdots & \vdots \\ v_{m-1,b} & v_{m-1,b+1} & \cdots & v_{m-1,mb} \end{pmatrix} \tag{12}$$

$$Q_4 = \begin{pmatrix} v_{1,mb-b+1} & \cdots & v_{1,mb-1} & v_{1,mb} \\ v_{1,mb-b+2} & \cdots & v_{1,mb} & * \\ \vdots & \vdots & \cdots & \vdots \\ v_{1,mb+1} & \cdots & * & * \\ v_{2,mb-b+1} & \cdots & v_{2,mb-1} & v_{2,mb} \\ v_{2,mb-b+2} & \cdots & v_{2,mb} & * \\ \vdots & \vdots & \cdots & \vdots \\ v_{2,mb+1} & \cdots & * & * \\ \vdots & \vdots & \cdots & \vdots \\ v_{m-1,mb-b+1} & \cdots & v_{m-1,mb-1} & v_{m-1,mb} \\ v_{m-1,mb-b+2} & \cdots & v_{m-1,mb} & * \\ \vdots & \vdots & \cdots & \vdots \\ v_{1,mb+1} & \cdots & * & * \end{pmatrix} \tag{13}$$

where $*$s are linear combinations of the entries of the previous row. Note that Q^{-1} can be got by permuting the rows of Q_P^{-1}. Since Q_P is invertible, $det(Q_P) = det(Q_3) = 1$.

Let Γ_k be the set of polynomial functions of $\mathcal{U}$ variables with degree k. We now proceed to analyse some of the minors of Q_P.

Lemma 1: For $1 \leq j \leq (mb-b)$, $\mu(Q_P[b,j])) \in \Gamma_{mb-b}$

Proof: For two matrices A and B with the same number of rows, let $[AB]_{p,q}$ be the matrix which is got by removing the p^{th} column from A and appending the q^{th} column of B to A. For $i = b$ and $1 \leq j \leq (mb-b)$, $\mu(Q_P[i,j]))$ is given by:

$$\mu(Q_P[i,j]) = det([Q_3Q_4]_{j,1}) \tag{14}$$

Recall that, for a binary matrix $M \in \mathbb{F}_2^{mb\times mb}$, its determinant is given by the following formula,

$$det(M) = \sum_{f\in S_{mb}} \prod_{1\leq i\leq n} M(i, f(i)) \tag{15}$$

where S_{mb} is the set of permutations on $(1, 2, \ldots, mb)$. Observe that the diagonal elements of $([Q_3Q_4])_{j,1}$ are distinct $v_{i,k}$s. Their product corresponds to the identity permutation in the determinant expansion formula for $[Q_3Q_4])_{j,1}$. The resultant monomial has degree $mb-b$. Further, this monomial will not occur as a result of any other permutation. Hence $det([Q_3Q_4]_{j,1})$ is always a polynomial of degree $mb-b$. □

Lemma 2: If $1 \leq i \leq b$ *then*

$$\mu(Q_P[i,j]) = \begin{cases} det(Q_3) & i+j = mb+1 \\ 0 & i+j >= mb+1 \end{cases} \tag{16}$$

Proof: Observe that, for $1 \leq i \leq b$ and $i+j = mb+1$, the $Q_P[i,j]$s are the anti-diagonal elements of Q_2. Clearly, the minors of these elements are all equal to the determinant of Q_3. As we have already seen, the invertibility of Q_P implies that this determinant is always 1. Therefore, $\mu(Q_P[i,j]) = 1$ when $i+j = mb+1$

Note that, for $1 \leq i \leq b$ and $i+j > mb+1$, the $Q_P[i,j]$s are the elements of Q_2 that are below the anti-diagonal. Observe that, if the row and column corresponding to such an element are removed from Q_P, then the first $b-1$ rows of the resulting matrix are always rank deficient. Therefore, the determinant of this matrix is always 0. Therefore, $\mu(Q_P[i,j]) = 0$. □

Lemma 3: If $b+1 \leq i \leq mb$ *and* $1 \leq j \leq n-b$, *then* $\mu(Q_P[i,j]) \in \Gamma_{mb-b-1}$.

Proof: Observe that the elements of Q_P considered in this lemma are elements of the sub-matrix Q_3. Therefore, $\mu(Q_P[i,j])$, for the range of i and j considered, is nothing but the determinant of the sub-matrix of Q_3 got by deleting the i^{th} row and j^{th} column of Q_3. The diagonal elements of such a sub-matrix are distinct $v_{i,j}s$. Their product will result in a monomial of degree $mb-b-1$. This corresponds to the identity permutation in the determinant expansion formula given by Equation 14. Observe that no other permutation generates this monomial. Hence, the minor will always have a monomial of degree $mb-b-1$. Therefore, $\mu(Q_P[i,j] \in \Gamma_{mb-b-1}$. □

Lemma 4: If $b+1 \leq i \leq n$ *and* $mb-b+1 \leq j \leq mb$, *then* $\mu(Q_P[i,j]) = 0$.

Proof: The elements of Q_P considered in this lemma are elements of the submatrix Q_4. Whenever the row and column corresponding to such an element is removed from Q_P, the rows of the submatrix Q_2 become linearly dependent. Therefore, the first b rows of the resultant matrix are always rank deficient. Consequently, $\mu(Q_P[i,j]) = 0$.

□

For a given matrix A with polynomial entries, let $\Theta(A)$ be the maximum degree among all the entries of A. As there are $mb-b$ rows in Q_P with variable entries, $\Theta(Q_P^{-1}) \leq mb-b$. Therefore, we get the following as a consequence of Lemma 1.

$$\Theta(Q^{-1}) = \Theta(Q_P^{-1}) = mb-b \tag{17}$$

Recall that the configuration matrix $\mathcal{C}$ is given by $QP_{mb}Q^{-1}$. We now use the above developed machinery to calculate $\Theta(\mathcal{C})$.

Theorem 1: $\Theta(\mathcal{C}) \geq mb - b$.

Proof: Observe that the gain matrices $B_0, B_1 \cdots, B_{b-1}$ appear in the last m rows of $C_{\mathcal{S}}$. These rows are generated by multiplying the last m rows QP_{mb} with Q^{-1}. The last m rows of QP_{mb} are as follows

$$\begin{pmatrix} 0 & 0 \cdots & 1 * \cdots * * \\ v_{1,b+1} & v_{1,b+2} \cdots & v_{1,mb} * \cdots * * \\ v_{2,b+1} & v_{2,b+2} \cdots & v_{2,mb} * \cdots * * \\ \vdots & \vdots \cdots & \vdots \vdots \cdots \vdots \vdots \\ v_{m-1,b+1} & v_{m-1,b+2} \cdots & v_{m-1,mb} * \cdots * * \end{pmatrix} \quad (18)$$

The element $C[mb-m+1, mb-m+1]$ is got by multiplying the $(mb-m+1)$-th row of QP_{mb} with the $(mb-m+1)$-th column of Q^{-1}. Note that the $(mb-m+1)$-th column of Q^{-1} is equal to the b-th column of Q_P^{-1}. As a consequence of Lemmas 1 and 2, this column has the following form.

$$Q^{-1}[:, mb-m+1] = (P_1, P_2, \cdots, P_{mb-b}, 1, 0, \cdots, 0)^T \quad (19)$$

where $P_1, P_2, \cdots, P_{mb-b} \in \Gamma(mb-b)$. Therefore,

$$\begin{aligned} &C[mb-b+1, mb-b+1] \\ &= (\underbrace{0, 0, \cdots, 1}_{(mb-b)\text{ entries}}, *, \cdots, *, *) \\ &\quad \times (P_1, P_2, \ldots, P_{mb-b}, 1, 0, \cdots, 0)^T \\ &= P_{mb-b} \end{aligned}$$

Hence, it is proved that $\Theta(C) \geq mb - b$. □

Example 2: Consider a primitive $\sigma-$LFSR with 4, 2-input 2-output delay blocks i.e. $m = 2$ and $b = 4$. Therefore $n = mb = 8$. The primitive polynomial for the companion matrix P_z is $f(x) = x^8 + x^4 + x^3 + x^2 + 1$. The corresponding matrix Q_P has the following submatrices.

$$Q_1 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix} \quad (20)$$

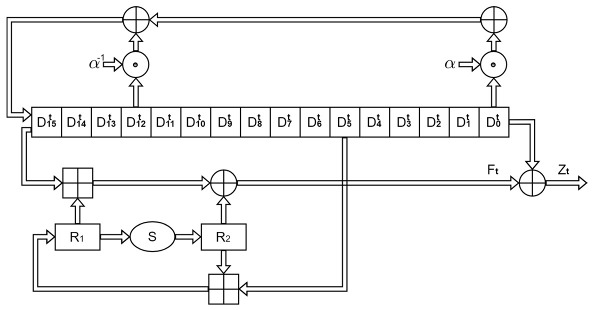
$$Q_2 = \begin{pmatrix} 0&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \\ 1&0&0&0 \end{pmatrix} \quad (21)$$

$$Q_3 = \begin{pmatrix} x_1&x_2&x_3&x_4 \\ x_2&x_3&x_4&x_5 \\ x_3&x_4&x_5&x_6 \\ x_4&x_5&x_6&x_7 \end{pmatrix} \quad (22)$$

$$Q_4 = \begin{pmatrix} x_5 & x_6 & x_7 & x_8 \\ x_6 & x_7 & x_8 & x_1+x_3+x_4+x_5 \\ x_7 & x_8 & x_1+x_3+x_4+x_5 & x_2+x_4+x_5+x_6 \\ x_8 & x_1+x_3+x_4+x_5 & x_2+x_4+x_5+x_6 & x_3+x_5+x_6+x_7 \end{pmatrix} \quad (23)$$

Therefore,

$$Q_P^{-1}[:, 4] = [Q^{-1}[:, 7] = (P_1, P_2, P_3, P_4, 1, 0, 0, 0)$$



α^{-1}
α
D^t_{15} D^t_{14} D^t_{13} D^t_{12} D^t_{11} D^t_{10} D^t_9 D^t_8 D^t_7 D^t_6 D^t_5 D^t_4 D^t_3 D^t_2 D^t_1 D^t_0
Ft
Zt
R1
S
R2

FIGURE 3. The block diagram of SNOW 2.0.

where

P_1 : $x_2x_4x_6x_8 + x_2x_4x_7 + x_2x_5x_8 + x_2x_6 + x_3x_6x_8 + x_3x_7 + x_4x_5x_6 + x_4x_6 + x_4x_8 + x_5$.

P_2 : $x_1x_4x_6x_8 + x_1x_4x_7 + x_1x_5x_8 + x_1x_6 + x_2x_3x_6x_8 + x_2x_3x_7 + x_2x_4x_5x_8 + x_2x_4x_6x_7 + x_2x_5x_6 + x_2x_5x_7 + x_3x_4x_8 + x_3x_5x_6 + x_3x_5x_8 + x_3x_6x_7 + x_4x_5 + x_4x_7$

P_3 : $x_1x_3x_6x_8 + x_1x_3x_7 + x_1x_4x_5x_8 + x_1x_4x_6x_7 + x_1x_5x_6 + x_1x_5x_7 + x_2x_3x_5x_8 + x_2x_3x_6x_7 + x_2x_4x_5x_7 + x_2x_4x_6 + x_2x_4x_8 + x_2x_5x_6 + x_2x_6x_8 + x_2x_7 + x_3x_4x_7 + x_3x_4x_8 + x_3x_5x_7 + x_3x_5 + x_4x_5 + x_4x_6$

P_4 : $x_1x_3x_5x_8 + x_1x_3x_6x_7 + x_1x_4x_5x_7 + x_1x_4x_6 + x_1x_4x_8 + x_1x_5x_6 + x_2x_3x_5x_7 + x_2x_3x_6 + x_2x_4x_7 + x_2x_5x_8 + x_2x_5 + x_2x_6x_7 + x_3x_4x_6 + x_3x_4x_7 + x_3x_8 + x_4x_5$

Here, $C_{\mathcal{S}}[7, 7]$, is equal to P_4 which is a polynomial of degree 4.

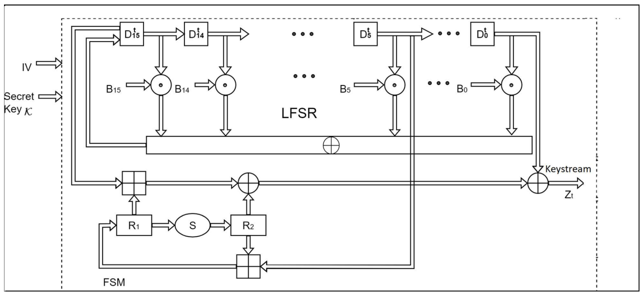
V. CASE STUDY: INTEGRATION WITH SNOW 2.0

In this subsection, we first introduce SNOW 2.0, and then use it as a case study to show how σ-KDFC can be applied to an LFSR-based cipher stream. We refer to the resulting cipher as KDFC-SNOW.

A. SNOW 2.0

The SNOW series of word based stream ciphers was first introduced in [7]. This version of SNOW is known as SNOW 1.0. This was shown to be vulnerable to a linear distinguishing attack as well as a guess and determine attack [29].

SNOW 2.0 (Adopted by ISO/IEC standard IS 18033-4) was introduced later in [8] as a modified version of SNOW 1.0. This version was shown to be vulnerable to algebraic and other attacks [17], [19], [28]–[31]. We consider SNOW 2.0 as a test case and demonstrate how replacing the LFSR in



IV
Secret Key $\mathcal{K}$
D^t_{15} D^t_{14} D^t_5 D^t_0
B_{15} B_{14} B_5 B_0
LFSR
Keystream
Z_t
R1
S
R2
FSM

FIGURE 4. The block diagram KDFC-SNOW.

this scheme with a σ-LFSR increases its resistance to various attacks.

The block diagram of SNOW 2.0 is shown in figure 3.

In figure 3, $+$ and $\boxplus$ represent bit wise XOR (addition in the field $GF(2)$) and integer addition modulo 2^{32} respectively. As shown in figure 3, the keystream generator in SNOW 2.0 consists of an LFSR and an FSM (Feedback State Machine). The LFSR implements the following linear recurring relation:

$$x_{n+16} = \alpha^{-1}x_{n+11} + x_{n+2} + \alpha x_0.$$

where, α is the root of the following primitive polynomial

$$G_S(x) = (x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239}) \in \mathbb{F}_{2^8}[X]$$

where β is the root of the following primitive polynomial.

$$H_S(x) = x^8 + x^7 + x^5 + x^3 + 1 \in \mathbb{F}_2[X]$$

The FSM contains two 32-bit registers $R1$ and $R2$. These registers are connected by means of an S-Box which is made using four AES S-boxes. This S-box serves as the source of nonlinearity.

B. KDFC-SNOW

In the proposed modification, we replace the LFSR part of SNOW 2.0 by a σ-LFSR having 16, 32-input 32-output delay blocks. The configuration matrix of the σ-LFSR is generated using Algorithm 1. We shall refer to the modified scheme, shown in Figure 4, as KDFC-SNOW.

During initialization the feedback function of the σ-LFSR is identical to that of SNOW-2. As in SNOW 2.0, the σ-LFSR is initialized using a 128-bit IV and a 128/256-bit secret key K. KDFC-SNOW is run with this configuration for 32 clock cycles without producing any symbols at the output. The vectors generated in the last 12 of these clock cycles are used in Algorithms 1 to generate a new feedback configuration. As we have already mentioned, some of the iterations of Algorithm 1 are pre-calculated and the remaining ones are done as a part of the initialization process. In this case, it is assumed that 468 of these iterations are pre-calculated and the last 12 iterations are carried out in the initialization process. This calculated configuration replaces the original one and the resulting set-up is used to generate the keystream.

C. INITIALIZATION OF KDFC-SNOW

- The delay blocks $D_0, \cdots, D_{15}$ are initialized using the 128/256 bit secret key K and a 128 bit IV in exactly the same manner as SNOW 2.0. The registers R_1 and R_2 are set to zero.
- The initial feedback configuration of the σ-LFSR is identical to SNOW 2.0. This is done by setting B_{11} and B_0 as matrices that represent multiplication by α^{-1} and α respectively. Further, B_2 is set to identity. The other gain matrices are set to zero.
- KDFC-SNOW is run in this configuration for 32 clock cycles without making the output externally available. The last 12 values of F^t are used as the random numbers in Algorithm 1.
- A new configuration matrix is calculated using Algorithms 1 and the corresponding feedback configuration replaces the original one. The scheme is now run with this configuration. The first 32 vectors are discarded and the key stream starts from the 33rd vector.

D. GOVERNING EQUATIONS OF KDFC-SNOW

Let $D_i^t \in \mathbb{F}_{2^{32}}$ denote the value stored in the i^{th} delay block at the t-th time instant after the key stream generation has started. The outputs of the delay blocks of the σ-LFSR are related as per the following equation:

$$D_{15}^{t+1} = B_0 D_0^t + B_1 D_1^t + \cdots + B_{15} D_{15}^t \tag{24}$$

Therefore,

$$D_k^{t+1} = \begin{cases} D_{k+t+1}^0 & 0 \le k+t+1 \le 15 \\ \sum_{i=0}^{15} B_i D_i^{t+k} & k+t+1 > 15 \end{cases} \tag{25}$$

The value of the keystream at the t-th time instant is given by the following equation Let F_t be the output of the FSM at time t,

$$F_t = z^t + D_0^t = (D_{15}^t \boxplus R_1^t) + R_2^t \tag{26}$$

The registers are updated as follows:

$$\begin{aligned} R_1^{t+1} &= D_5^t \boxplus R_2^t && (27)\\ R_2^{t+1} &= S(R_1^t) && (28)\\ z^t &= R_1^t \boxplus D_{15}^t + R_2^t + D_0^t \\ &= (R_2^{t-1} \boxplus D_4^t) \boxplus D_{15}^t + R_2^t + D_0^t && (29) \end{aligned}$$

where R_1^t and R_2^t represent the values of registers R_1 and R_2 at time instant t. The operation "$\boxplus$" is defined as follows:

$$x \boxplus y = (x+y) \mod 2^{32} \tag{30}$$

The challenge for an adversary in this scheme is to find the gain matrices $\{B_0, B_1, \cdots, B_{15}\}$ in addition to the initial state $\{D_0^0, \cdots, D_{15}^0\}$.

Note that Equations 26 to 29 are got from the FSM. Since the FSM part of the keystream generator is identical for SNOW 2.0 and KDFC-SNOW these equations are identical for both schemes.

E. SECURITY ENHANCEMENT DUE TO KDFC-SNOW

The primary objective of the proposed scheme is to resist Algebraic Attack. In these section, we describe how this happens. Further, for completeness, we briefly describe the performance of KDFC-SNOW against other attacks such as Distinguishing Attack, Fast Correlation Attack and Guess and Determining Attack.

1) ALGEBRAIC ATTACK:

We first briefly the Algebraic attack on SNOW 2 described in [17] and demonstrate why this attack becomes difficult with KDFC-SNOW. This attack first attempts to break a modified version of the scheme where the $\boxplus$ operator is approximated by $\oplus$. The state of LFSR and the value of the registers at the end of the 32 initialization cycles are considered unknown variables. This accounts for a total of $512 + 32 = 544$ unknown variables. The algebraic degree of each of the S-box(S) equations (156 linearly independent quadratic equations in each clock cycle) is 2. Rearranging the terms in Equation 29, we get the following

$$R_2^t = (R_2^{t-1} \boxplus D_4^t) \boxplus D_{15}^t + D_0^t + z^t. \tag{31}$$

Note that $R_1^0 = R_2^0 + z^0 + D_0^0 + D_{15}^0$.Therefore, by approximating $\boxplus$ as $\oplus$, Equation 31 expands to the following:

$$R_2^t = R_2^0 + \sum_{i=0}^{t} z^i + \sum_{i=0}^{t}(D_4^i + D_{15}^i + D_0^i) \tag{32}$$

Further, Equation 28 can be expanded as follows:

$$R_2^{t+1} = S(R_1^t) = S(R_2^t + z^t + D_{15}^t + D_0^t) \tag{33}$$

In Equation 33, the outputs of the delay blocks can be related to the initial state of the LFSR using the following equation.

$$D_k^{t+1} = \begin{cases} D_{k+t+1}^0 & 0 \leq k+t+1 \leq 15 \\ \alpha^{-1}D_{11}^{k+t} + D_2^{k+t} + \alpha D_0^{k+t} & k+t+1 > 15 \end{cases} \tag{34}$$

Because of the nature of the S-Box, Equation 33 gives rise to 156 quadratic equations per time instant ([17]). When these equations are linearized, the number of variables increases to $\sum_{i=0}^{2}\binom{544}{i} \approx 2^{17}$. Therefore with $2^{17}/156 \approx 951$ samples, we get a system of equations, which can be solved in $\mathcal{O}(2^{51})$ time, to obtain the initial state of the LFSR and the registers. This attack is then modified to consider the $\boxplus$ operator. This attack has a time complexity of approximately $\mathcal{O}(2^{294})$.

When the LFSR in SNOW 2.0 is replaced by a σ-LFSR, the feedback equation is no longer known. If the entries of the feedback gain matrices are considered as unknowns, then there are a total of $16 * m^2 + mb + m = 16928$ unknown variables (This includes the $16 * m^2$ entries of the feedback matrices and $mb + m$ entries corresponding to the state of the LFSR and the register R_2 at the beginning of the key stream). The output of the delay blocks at any given instant are functions of these variables. Here, Equation 34 is replaced by Equation 25. Now, if the outputs of the delay blocks in Equation 33 are linked to the initial state of the LFSR (i.e. the state when the key stream begins) using Equation 25 instead of Equation 34, then the resulting equations contain the feedback matrices and their products. For example, in the expressions for R_2^2 and R_2^3, D_{15}^1 and D_{15}^2 are given as follows

$$\begin{aligned} D_{15}^1 &= B_0D_0^0 + B_1D_1^0 + \ldots + B_{15}D_{15}^0 \\ D_{15}^2 &= B_0D_0^1 + B_1D_1^1 + \ldots + B_{15}D_{15}^1 \\ &= B_0D_1^0 + B_1D_2^0 + \ldots \\ &\quad +B_{15}(B_0D_0^0 + B_1D_1^0 + \ldots + B_{15}D_{15}^0) \\ &= B_{15}B_0D_0^0 + (B_{15}B_1 + B_0)D_1^0 + \ldots \\ &\quad +(B_{15}B_{15} + B_{14})D_{15}^0 \end{aligned}$$

Observe that, while D_{15}^1 is a polynomial of degree two in the unknown variables, D_{15}^2 is a polynomial of degree 3. Similarly, with each successive iteration the degree of the expression for D_{15}^t keeps increasing, till all the m^2b entries of the feedback matrices are multiplied with each other. A similar thing happens with the expressions for D_0^t. This results in a set of polynomial equations having maximum degree equal to $m^2b + 1 = 16385$. Therefore, although the equations generated by Equation 33 are quadratic in terms of the initial state of the σ-LFSR, they are no longer quadratic in the set of all unknowns. We instead have a system of equations in 16982 variables with a maximum degree of degree of over 16000. Linearizing such a system will give us a system of linear equations in $N = \sum_{i=0}^{M}\binom{16928}{i}$ unknowns where M is higher than 16000. Such an attack is therefore not feasible.

One could instead consider the rows of the matrix Y generated by Algorithm 1 as unknowns. Assuming that the first row is e_1^n, the total number of unknowns will now be $31 * 512 = 15872$. As we have already seen, the entries of the feedback matrices (B_is) are polynomials in these variables. From Theorem 1, the maximum degree of these polynomials is atleast $mb - b$. Therefore, the maximum degree of the equations generated by Equation 33 will be atleast $mb - b + 1 = 497$. Therefore, linearizing this system of equations gives rise to a system of linear equations in $N = \sum_{i=0}^{497}\binom{16416}{i} \approx \mathcal{O}(2^{3207})$ unknowns. Therefore, an algebraic attack on this scheme that uses linearization seems unfeasible.

Another approach to the algebraic cryptoanalysis of SNOW 2 is found in [32]. Therein, linear equations are generated by assuming a set of values for the entries of the registers of the FSM. Two methods of cryptoanalysis are presented in this paper. In the first method, the attacker guesses the values of ten consecutive entries of the register R_1. The guessed value of R_1^t uniquely determines the value of $= R_2^{t+1}$ (by Equation 28). Given values of R_1^t and R_2^t, Equation 26 gives rise to two linear equations and thirty quadratic equations. These equations can be generated for nine time instances. Further, given values of R_1^{t+1} and R_2^t, the value of D_5^t can be uniquely determined. One can thus determine 8 consecutive values of D_5^t. These along with the linear and quadratic equations results in an over-determined system of equations. These, when solved, gives us an estimate of the current state. Since the relation between the current state and the initial state is linear, one can get an estimate of the initial state from the estimate of the current state. The correctness of the guesses can be verified by using the estimated initial state and the system equations of SNOW 2.0 to regenerate the key stream and check if it matches with the actual one.

In the second method, the attacker assumes the following nine consecutive entries of the register R_1

$$R_1^t = 0, \quad R_1^{t+1} = 2^{32} - 1, \; R_1^{t+2} = 0, \cdots, R_1^{t+8} = 0 \tag{35}$$

Now, there are 7 consecutive time instances,k, where the values of R_1^{k+1} and R_2^k are simultaneously known. Hence, at each of these time instances, Equation 27 gives rise to 32 linear equations over $GF(2)$. This accounts for a total of 224 equations. Further, there are 7 consecutive values of k wherein the value of R_1^k is 0 and the value of R_2^k is known. Therefore, Equation 26 gives us 32 linear equations over $GF(2)$ at each of these time instances. This accounts for another 224 linear equations. If the assumption made in this attack holds true, at the $t+1$-th time instance the value in the register R_1 is zero while the value in the regiester R_2 is given as follows.

$$R_2^{t+1} = D_5^{t+1} + 1111\cdots 1 \tag{36}$$

When these values are substituted in Equation 26, we get another 32 linear equations. Recall that, at the $t+2$-th time instance, the value in the register R_1 is $1111\cdots 1$. As a consequence of Theorem 2 in [32], this results in Equation 26 generating 32 linear equations which are satisfied with probability half. This probability becomes 1 when $D_0^2 + z^2 + S(0)$ is zero. We thus have a total of 512 linear equations. When these equations are linearly independent, solving them gives us the state of the LFSR. The correctness of the assumptions is verified by checking if the sequence generated from this state matches with the actual keystream.

In both these attacks, in order to verify the correctness of the assumptions, one has to generate the sequence with the calculated state of the LFSR and check if it matches the actual keystream. To do this, the feedback equation of the LFSR is needed. Since this is not available in KDFC-SNOW, this verification cannot be done. As a result, KDFC-SNOW is immune to these attacks. For a similar attack to work on KDFC-SNOW, the assumptions should enable the attacker to calculate 512 output words of the LFSR (as against the 32 output words that are calculated in these attacks). This would mean more assumptions. Consequently, the probability of these assumptions being true will be significanlty lower. This will result in a much higher time complexity for the attack.

2) DISTINGUISHING ATTACK:

In the distinguishing attack, the attacker aims to distinguish the generated keystream from a random sequence. Distinguishing attacks on SNOW 2.0 have been launched using the linear masking method [18]–[20]. This method essentially adapts the linear cryptanalysis method given in [33] to stream ciphers. In this method, the algorithm of the key stream generator is assumed to consist of two parts, a linear one and a non-linear one. In the case of SNOW 2.0, the linear part is the LFSR and the non-linear part is the FSM. The linear part satisfies a linear recurring relation of the form $f(x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+k}) = 0$ for all n. We then try to find a linear relation, called the masking relation, that the non-linear part approximately satisfies. This relation is of the following form:

$$\sum_{i=0}^{\ell_1} \Gamma_i x_{n+i} = \sum_{i=0}^{\ell_2} \Lambda_i z_{n+i} \tag{37}$$

where $z_0, z_1, \ldots$ is the output sequence of the key stream generator. The Γ_is and Λs are linear masks that map the corresponding x_{n+i}s and z_{n+i}s to $\mathbb{F}_2$ respectively. The error in the masking relation can be seen as a random variable. If p is the probability that the non linear part satisfies Equation 37, then $p - \frac{1}{2}$ is called the bias of the masking relation. The Masking relation along with the linear recurring relation is used to generate a relation in terms of the elements of the output sequence. The error in this relation can also be seen as a random variable. If the probability of the sequence satisfying this relation is p_f, then $p_f - \frac{1}{2}$ is the bias of this relation. This bias can be related to the bias of the masking relation using the piling up lemma in [33]. The main task in this type of attack is to find masks Γ_is and Γ_i's which maximise the bias of the masking relation. The following linear masking equation is used in [18] and [19] for the FSM of SNOW 2.

$$\begin{aligned}\Gamma_0 x_n + \Gamma_1 x_{n+1} + \Gamma_5 x_{n+5} + \Gamma_{15} x_{n+15} + \Gamma_{16} x_{n+16} \\ = \Lambda_0 z_n + \Lambda_1 z_{n+1}\end{aligned}$$

In [18] it is assumed that all the Γ_is and Λ_is are equal to each other. In [19] it is assumed that Γ_0, Γ_{15} and Λ_0 are equal to each other. Γ_1, $\Gamma_5.\Gamma_{16}$ and Λ_1 are also assumed to be equal. Since $f(x_n, x_{n+1}, \ldots, x_{n+k})$ is a linear relation, the following relation can be written purely in terms of the z_is

$$\begin{aligned}&\Gamma_0 f(x_n, x_{n+1}, \ldots, x_{n+k})\\ &\quad + \Gamma_1 f(x_{n+1}, x_{n+2}, \ldots, x_{n+k+1}\\ &\quad + \Gamma_5 f(x_{n+5}, x_{n+6}, \ldots, x_{n+k+5})\\ &\quad + \Gamma_{15} f(x_{n+15}, x_{n+16}, \ldots, x_{n+k+15})\\ &\quad + \Gamma_{16} f(x_{n+16}, x_{n+17}, \ldots, x_{n+k+16}) = 0\end{aligned}$$

The linear relation between the elements of the output sequence in both [18] and [19] is obtained using this method. Further, if there are ℓ non-zero coefficients in f, then the random variable corresponding to the error in this relation is a sum of ℓ random variables each corresponding to the error in the linear masking equation.

In the proposed σ-LFSR configuration, the feedback equation is not known. Therefore the only known linear recurring relation that the output of the σ-LFSR satisfies is the one defined by its characteristic polynomial. If the characteristic polynomial is assumed to be the same as that of the LFSR in SNOW 2, then the corresponding linear recurring relation has 250 non-zero coefficients. Further, since these coefficients are elements of $\mathbb{F}_2$, the non-zero coefficients are all equal to 1. Therefore, as a consequence of the piling up lemma, if the bias of the masking equation is ϵ, then the bias of the relation between the elements of the key stream is given as follows

$$\epsilon_{final} = 2^{249} \times \epsilon^{250} \tag{38}$$

The number of elements of the key stream needed to distinguish it from a random sequence is $\frac{1}{\epsilon_{final}^2}$. Therefore, for an identical linear masking equation, the length of the key stream for the distinguishing attack is much higher for the proposed configuration as compared to SNOW 2. This is demonstrated in the following table.

3) FAST CORRELATION ATTACK

The Fast Correlation Attack is a commonly used technique for the cryptanalysis of LFSR based stream ciphers. This method was first introduced for bitwise keystreams in ([34]). Here, the attacker views windows of the key stream as noisy linear encodings of the initial state of the LFSR. She then tries to recover the initial state by decoding this window. Further, linear combinations of elements in this window can be seen as encodings of subsets of the initial state. This results in smaller codes which are more efficient to decode [35]. The linear recurring relation satisfied by the output of the LFSR is used to generate the parity check matrix for this code. A Fast correlation attack for word based stream ciphers was first described in [36]. An improvement on this attack is given in [20]. Both these schemes consider a linear recurring relation with coefficients in $\mathbb{F}_2$. For SNOW 2.0, this relation has order 512. This is equivalent to considering each component sequence to be generated by a conventional bitwise LFSR having the same characteristic polynomial as the LFSR in SNOW 2.0. The time complexity of the attack given in [20] is $2^{212.38}$. The scheme in [21] considers the LFSR in SNOW 2.0 to be over $\mathbb{F}_{2^8}$. This results in a linear recurring relation of order 64. Further, it utilizes the $k-$ tree Algorithm given in [37] to generate parity check equations. This results in a significant improvement in the time complexity of the attack. The time complexity of this attack is $2^{164.5}$ which is around 2^{49} times better than that of the attack given in [20]. However, in order to derive the linear recurring relation over $\mathbb{F}_{2^8}$, the knowledge of the feedback function is critical. In KDFC SNOW, the characteristic polynomial of the σ-LFSR is publicly known. The attacker can therefore generate a linear recurring relation over $\mathbb{F}_2$ that the output of the σ-LFSR satisfies. Therefore, the attack given in [20] will also be effective against KDFC-SNOW. However, without the knowledge of the feedback function, the attacker will not be able to derive a linear recurring relation over $\mathbb{F}_{2^8}$. Hence, KDFC-SNOW is resistant against the attack given in [21].

[38] uses MILP(Mixed Integer Linear Programming) to find a linear mask that gives better correlation. This results in an attack with a time complexity of $2^{162.91}$ which is $2^{1.59}$ times better than [21]. [39] further modifies this attack using a small trick in $k-$ tree algorithm. The time complexity with this modification turns out to be $2^{162.86}$. These attacks consider a feedback polynomial of degree 512 over $\mathbb{F}_2$. Therefore, KDFC-SNOW does not provide any extra security against these attacks.

4) GUESS AND DETERMINE ATTACK

In a Guess and Determine Attack, the attacker aims to estimate the values of a minimum number of variables using which the complete sequence can be constructed. For SNOW 2.0, this includes the values of the outputs of the delay blocks of the LFSR and the outputs of the registers of the FSM at some time instant. This is done by guessing some of the values and determining the rest of them using system equations. If the sequence generated using these estimates matches the output of the key-stream generator, then the guesses are deemed to be correct. Otherwise, a fresh set of guesses are considered. The set of variables whose values are guessed is known as the basis for the attack. For both SNOW 2.0 and KDFC-SNOW, these variables take their values from $\mathbb{F}_2^{32}$. Hence, if the size of the basis is k, then the probability of a correct guess is 2^{-32k}. Thus, on average, one needs $\mathcal{O}(2^{-32k})$ attempts to make a correct guess. Therefore, the problem here is to find a basis of the minimum possible size. A systematic Vitterbi-like algorithm for doing this is given in [28].The complexity of this attack was found to be 2^{265}([28]) for SNOW2.0. The complexity of this attack reduced to 2^{192} in ([31]) by incorporating a couple of auxiliary equations. We now briefly describe the algorithm given in [28] in the context of SNOW 2.0

TABLE 2. Comparison of Distinguishing Attack Result for SNOW 2.0 and KDFC SNOW.

Reference	ϵ	ϵ_{final} SNOW 2.0	ϵ_{final} KDFC SNOW	**#Keystream** SNOW 2.0	**#Keystream** KDFC-SNOW
[18]	$2^{-27.61}$	$2^{-112.25}$	$2^{-6653.5}$	2^{225}	2^{13307}
[19]	$2^{-15.496}$	$2^{-86.9}$	2^{-3625}	2^{174}	2^{7250}

Consider the following equations which are satisfied by SNOW 2.0

$$D_t^{16} = \alpha^{-1}D_t^{11} + D_t^2 + \alpha D_t^0 \tag{39}$$

$$R1_t = D_t^4 + S(R1_{t-2}) \tag{40}$$

$$z_t = D_t^0 + (D_t^{15} + R1_t) + S(R1_{t-1}) \tag{41}$$

These equations are used to generate the following tables The entries in the above tables correspond to the variables that are to be estimated. The entries in the first table, i.e. 0 to 34, correspond to 35 consecutive outputs of the LFSR. The entries 35 to 55 correspond to 21 consecutive entries of Register $R1$. Each row of the above tables correspond to the values of the delay blocks and registers in Equations 39, 40 and 41 at a particular time instant.

We now consider a multi-stage graph with 56 nodes in each stage corresponding to the 56 entries in the above tables. Each node is connected to all the nodes in the next stage giving rise to a trellis diagram. An entry is said to be eliminated by a path if, knowing the values of the entries corresponding to the nodes in the path, the value of that entry can be calculated.

We now recursively calculate an optimal path that eliminates all the entries. The desired basis corresponds to the nodes in this path. In the i-th iteration of this algorithm, we calculate the optimal path of length i to each node in the i-th stage. In order to find the optimal path to the k-th node, we consider all the incoming edges of node k. By appending the node k to the optimal paths of length $i - 1$ ending at the source nodes of these edges, we get 55 paths of length i. We choose the edge corresponding to the path that eliminates the most number of variables. In case of a tie, we consider the path that results in the most number of rows with 2 unknowns and so on. This process is continued till we get a path that eliminates all the entries. This algorithm results in a basis of cardinality 8 for SNOW 2.0.

In KDFC-SNOW, the feedback equation of the σ-LFSR is not known. The smallest known linear recurring relation that the output of the σ-LFSR satisfies is the relation corresponding to its characteristic polynomial. This relation is given as follows

$$\begin{aligned}
x_{n+512} = {} & x_{n+510} + x_{n+504} + x_{n+502} + x_{n+501} + x_{n+494} \\
& + x_{n+493} + x_{n+490} + x_{n+486} + x_{n+485} + x_{n+483} \\
& + x_{n+481} + x_{n+480} + x_{n+478} + x_{n+477} + x_{n+471} \\
& + x_{n+470} + x_{n+469} + x_{n+466} + x_{n+462} + x_{n+461} \\
& + x_{n+459} + x_{n+458} + x_{n+452} + x_{n+449} + x_{n+446} \\
& + x_{n+445} + x_{n+444} + x_{n+441} + x_{n+438} + x_{n+437} \\
& + x_{n+434} + x_{n+433} + x_{n+432} + x_{n+431} + x_{n+429} \\
& + x_{n+427} + x_{n+424} + x_{n+423} + x_{n+420} + x_{n+419} \\
& + x_{n+414} + x_{n+412} + x_{n+411} + x_{n+409} + x_{n+405} \\
& + x_{n+402} + x_{n+400} + x_{n+399} + x_{n+398} + x_{n+396} \\
& + x_{n+395} + x_{n+393} + x_{n+392} + x_{n+390} + x_{n+388} \\
& + x_{n+387} + x_{n+385} + x_{n+375} + x_{n+374} + x_{n+372} \\
& + x_{n+371} + x_{n+366} + x_{n+365} + x_{n+363} + x_{n+362} \\
& + x_{n+359} + x_{n+357} + x_{n+356} + x_{n+355} + x_{n+354} \\
& + x_{n+353} + x_{n+352} + x_{n+351} + x_{n+350} + x_{n+347} \\
& + x_{n+345} + x_{n+344} + x_{n+343} + x_{n+341} + x_{n+339} \\
& + x_{n+338} + x_{n+337} + x_{n+336} + x_{n+333} + x_{n+330} \\
& + x_{n+329} + x_{n+326} + x_{n+324} + x_{n+322} + x_{n+319} \\
& + x_{n+310} + x_{n+307} + x_{n+306} + x_{n+305} + x_{n+304} \\
& + x_{n+303} + x_{n+301} + x_{n+299} + x_{n+298} + x_{n+297} \\
& + x_{n+296} + x_{n+295} + x_{n+294} + x_{n+293} + x_{n+292} \\
& + x_{n+291} + x_{n+289} + x_{n+286} + x_{n+285} + x_{n+283} \\
& + x_{n+282} + x_{n+281} + x_{n+278} + x_{n+276} + x_{n+274} \\
& + x_{n+271} + x_{n+269} + x_{n+264} + x_{n+262} + x_{n+259} \\
& + x_{n+258} + x_{n+257} + x_{n+255} + x_{n+253} + x_{n+251} \\
& + x_{n+249} + x_{n+248} + x_{n+243} + x_{n+240} + x_{n+239} \\
& + x_{n+238} + x_{n+236} + x_{n+235} + x_{n+233} + x_{n+232} \\
& + x_{n+230} + x_{n+229} + x_{n+228} + x_{n+227} + x_{n+226} \\
& + x_{n+222} + x_{n+217} + x_{n+216} + x_{n+215} + x_{n+214} \\
& + x_{n+213} + x_{n+210} + x_{n+208} + x_{n+206} + x_{n+203} \\
& + x_{n+201} + x_{n+199} + x_{n+193} + x_{n+190} + x_{n+184} \\
& + x_{n+179} + x_{n+178} + x_{n+177} + x_{n+175} + x_{n+174} \\
& + x_{n+173} + x_{n+172} + x_{n+171} + x_{n+169} + x_{n+165} \\
& + x_{n+164} + x_{n+163} + x_{n+158} + x_{n+156} + x_{n+155} \\
& + x_{n+153} + x_{n+152} + x_{n+151} + x_{n+149} + x_{n+147} \\
& + x_{n+146} + x_{n+143} + x_{n+141} + x_{n+138} + x_{n+136} \\
& + x_{n+132} + x_{n+131} + x_{n+129} + x_{n+128} + x_{n+126} \\
& + x_{n+125} + x_{n+124} + x_{n+123} + x_{n+121} + x_{n+120} \\
& + x_{n+119} + x_{n+118} + x_{n+117} + x_{n+116} + x_{n+115} \\
& + x_{n+113} + x_{n+112} + x_{n+111} + x_{n+109} + x_{n+105} \\
& + x_{n+104} + x_{n+103} + x_{n+102} + x_{n+98} + x_{n+97} \\
& + x_{n+94} + x_{n+93} + x_{n+89} + x_{n+88} + x_{n+87} \\
& + x_{n+81} + x_{n+78} + x_{n+76} + x_{n+75} + x_{n+73} \\
& + x_{n+72} + x_{n+70} + x_{n+69} + x_{n+68} + x_{n+67} \\
& + x_{n+66} + x_{n+65} + x_{n+63} + x_{n+59} + x_{n+58} \\
& + x_{n+57} + x_{n+56} + x_{n+55} + x_{n+53} + x_{n+51} \\
& + x_{n+50} + x_{n+49} + x_{n+47} + x_{n+46} + x_{n+45} \\
& + x_{n+44} + x_{n+41} + x_{n+39} + x_{n+37} + x_{n+36} \\
& + x_{n+33} + x_{n+30} + x_{n+26} + x_{n+25} + x_{n+21} \\
& + x_{n+20} + x_{n+19} + x_{n+16} + x_{n+5} + x_0
\end{aligned}$$

TABLE 3. Index table for SNOW 2.0.

0	2	11	16
1	3	12	17
⋮	⋮	⋮	⋮
18	20	29	34

4	35	37
5	36	38
⋮	⋮	⋮
22	53	55

0	15	36	37
1	16	37	38
⋮	⋮	⋮	⋮
18	33	54	55

As in SNOW 2.0, the following equations are also satisfied,

$$\begin{aligned}
R1_n &= x_{n+4} + S(R1_{n-1}) \\
z_n &= x_n + (x_{n+15} + R1_n) + S(R1_{n-1})
\end{aligned}$$

The following tables can be constructed using these three equations. We ran the Vitterbi-like algorithm with the above tables on a cluster with 40 INTEL(R) XEON(R) CPUs (E5-2630 2.2GHz). The program ran for 16 iterations and generated the path $\{1041, 17, 15, 13, 28, 16, 11, 14, 9, 1050, 18, 39, 12, 7, 5, 0, 3\}$. This path has length 17. This corresponds to a time complexity of $\mathcal{O}(2^{544})$.

5) CACHE TIMING ATTACK

It is a kind of side-channel analysis attack where the cache memory is accessed by the adversary before or after the generation of each keystream bit. In schemes like SNOW 2.0 (and other schemes in the SNOW series), multiplication in the finite field is performed using look up tables corresponding to the non-zero constants in the feedback equation. Further, another look up table is used for the implementation of the S-BOX. These implementations are cache friendly. However, the adversary can extract secret information about the LFSR by monitoring the cache access. According to the attack model of [40], [41], the adversary uses two synchronous oracles:

1 KEYSTREAM(J) It returns the $J-$th keystream block.
2 SCA-KEYSTREAM(J) It returns the unordered list of cache accesses which is done during the creation of the $J-$th keystream.

In SNOW 2.0 and SNOW 3G there are multiplications over $\mathbb{F}_{2^{32}}$ that are done in every clock cycle viz. multiplication by α and α^{-1}. These multiplications are implemented as follows

$$\alpha * x = (x \ll 8) \oplus T_1[x_3] \tag{42}$$

$$\alpha^{-1} * x = (x \gg 8) \oplus T_2[x_0] \tag{43}$$

where $x = (x_3x_2x_1x_0) \in F_{2^{32}}$ and T_1, T_2 are two 8×32 tables. Thus, from the list of cache accesses the adversary can extract 8-bits of information viz. the first four bits of S_t and the last four bits of S_{t+11}. The adversary then gathers all linear equations formed from these bits for $(512/8 = 64)$ clock cycles. The state of the LFSR can then be found by solving these linear equations.

In the proposed KDFC scheme, the feedback gains do not correspond to known elements of the finite field. Instead, the feedback gains are matrices which are dependent on the key. Therefore, the feedback equation cannot be calculated by using look up tables. As a result, the attacker cannot obtain any information from the list of cache accesses.

F. RANDOMNESS TEST

In this subsection, we evaluate the randomness of the keystream generated by KDFC-SNOW.

1) TEST METHODOLOGY

We have used the NIST randomness test suite to evaluate the randomness of a keystream generated by KDFC-SNOW. There are 16 randomness tests in the suite. Each test returns a level of significance i.e. $P - Value$. If this value is above 0.01 for a given test, then the keystream is considered to be random for that test.

KDFC-SNOW has been implemented using SageMath 8.0. The NIST randomness tests have been conducted on the generated keystream using Python 3.6. The characterestic polynomial of the σ-LFSR has been taken as

$$\begin{aligned}
f(x) = {} & x^{512} + x^{510} + x^{504} + x^{502} + x^{501} + x^{494} + x^{493} \\
& + x^{490} + x^{486} + x^{485} + x^{483} + x^{481} + x^{480} + x^{478} \\
& + x^{477} + x^{471} + x^{470} + x^{469} + x^{466} + x^{462} + x^{461} \\
& + x^{459} + x^{458} + x^{452} + x^{449} + x^{446} + x^{445} + x^{444} \\
& + x^{441} + x^{438} + x^{437} + x^{434} + x^{433} + x^{432} + x^{431} \\
& + x^{429} + x^{427} + x^{424} + x^{423} + x^{420} + x^{419} \\
& + x^{414} + x^{412} + x^{411} + x^{409} + x^{405} + x^{402} + x^{400} \\
& + x^{399} + x^{398} + x^{396} + x^{395} + x^{393} + x^{392} + x^{390} \\
& + x^{388} + x^{387} + x^{385} + x^{375} + x^{374} + x^{372} + x^{371} \\
& + x^{366} + x^{365} + x^{363} + x^{362} + x^{359} + x^{357} + x^{356} \\
& + x^{355} + x^{354} + x^{353} + x^{352} + x^{351} + x^{350} + x^{347} \\
& + x^{345} + x^{344} + x^{343} + x^{341} + x^{339} + x^{338} + x^{337} \\
& + x^{336} + x^{333} + x^{330} + x^{329} + x^{326} + x^{324} + x^{322} \\
& + x^{319} + x^{310} + x^{307} + x^{306} + x^{305} + x^{304} + x^{303} \\
& + x^{301} + x^{299} + x^{298} + x^{297} \\
& + x^{296} + x^{295} + x^{294} + x^{293} + x^{292} + x^{291} + x^{289} \\
& + x^{286} + x^{285} + x^{283} + x^{282} + x^{281} + x^{278} + x^{276} \\
& + x^{274} + x^{271} + x^{269} + x^{264} + x^{262} + x^{259} + x^{258} \\
& + x^{257} + x^{255} + x^{253} + x^{251} + x^{249} + x^{248} + x^{243} \\
& + x^{240} + x^{239} + x^{238} + x^{236} + x^{235} + x^{233} + x^{232} \\
& + x^{230} + x^{229} + x^{228} + x^{227} + x^{226} + x^{222} + x^{217} \\
& + x^{216} + x^{215} + x^{214} + x^{213} + x^{210} + x^{208} + x^{206} \\
& + x^{203} + x^{201} + x^{199} + x^{193} + x^{190} + x^{184} + x^{179} \\
& + x^{178} + x^{177} + x^{175} + x^{174} + x^{173} + x^{172} + x^{171} \\
& + x^{169} + x^{165} + x^{164} + x^{163} + x^{158} + x^{156} + x^{155} \\
& + x^{153} + x^{152} + x^{151} + x^{149} + x^{147} + x^{146} + x^{143} \\
& + x^{141} + x^{138} + x^{136} + x^{132} + x^{131} + x^{129} + x^{128} \\
& + x^{126} + x^{125} + x^{124} + x^{123} + x^{121} + x^{120} + x^{119} \\
& + x^{118} + x^{117} + x^{116} + x^{115} + x^{113} + x^{112} + x^{111} \\
& + x^{109} + x^{105} + x^{104} + x^{103} + x^{102} + x^{98} + x^{97} \\
& + x^{94} + x^{93} + x^{89} + x^{88} + x^{87} + x^{81} + x^{78} + x^{76} \\
& + x^{75} + x^{73} + x^{72} + x^{70} + x^{69} + x^{68} + x^{67} + x^{66} \\
& + x^{65} + x^{63} + x^{59} + x^{58} + x^{57} + x^{56} + x^{55} + x^{53} \\
& + x^{51} + x^{50} + x^{49} + x^{47} + x^{46} + x^{45} + x^{44} + x^{41} \\
& + x^{39} + x^{37} + x^{36} + x^{33} + x^{30} + x^{26} + x^{25} + x^{21} \\
& + x^{20} + x^{19} + x^{16} + x^{5} + 1.
\end{aligned}$$

TABLE 4. Index table for $f1(x)$.

512	510	504	502	$\cdots$	5	0
513	511	505	503	$\cdots$	6	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1025	1023	1007	1005	$\cdots$	516	513

TABLE 5. Index table for Equation 31 and Equation 32.

4	1026	1028
5	1027	1029
$\vdots$	$\vdots$	$\vdots$
517	1539	1541

0	15	1027	1028
1	16	1028	1029
$\vdots$	$\vdots$	$\vdots$	$\vdots$
513	528	1540	1541

(This polynomial is the characteristic polynomial of the LFSR in SNOW 2.0 when it is implemented as a σ-LFSR i.e. when multiplication by α and α^{-1} are represented by matrices).

The keystream has been generated using the following key (K) and initialization vector (IV).

$$K = [681, 884, 35, 345, 203, 50, 912, 358]$$

$$IV = [645, 473, 798, 506]$$

2) TEST RESULTS

The results obtained from 16 NIST tests are shown in table 6.

These results are comparable to that of SNOW 2.0([42]). Note that the feedback configuration of SNOW 2.0 is one of the possible feedback configurations in the σ-KDFC scheme.

TABLE 6. NIST Randomness Test.

S.No.	Test	P-Value	Random
01.	Frequency Test (Monobit)	0.35966689490586123	✓
02.	Frequency Test within a Block	0.24374184001729746	✓
03.	Run Test	0.9038184342313019	✓
04.	Longest Run of Ones in a Block	0.5246846287441829	✓
05.	Binary Matrix Rank Test	0.1371167998339736	✓
06.	Discrete Fourier Transform (Spectral) Test	0.1371167998339736	✓
07.	Non-Overlapping Template Matching Test	0.3189818228443801	✓
08.	Overlapping Template Matching Test	0.211350493609367	✓
09.	Maurer's Universal Statistical test	0.4521082097311434	✓
10.	Linear Complexity Test	0.1647939201114819	✓
11.	Serial Test	0.7821664366290292	✓
12.	Approximate Entropy Test	0.880218270580662	✓
13.	Cummulative Sums (Forward) Test	0.34630799549695923	✓
14.	Cummulative Sums (Reverse) Test	0.6633686090204551	✓
15.	Random Excursions Test	0.969735280059932	✓
16.	Random Excursions Variant Test	0.97387	✓

G. CHALLENGES IN IMPLEMENTATION

The feedback function of a σ-LFSR can be implemented in hardware by ANDing the bits at the output of the delay blocks with the corresponding columns of the feedback matrices and then EXORing their respective outputs. This can be implemented by a fairly fast combinational circuit.

The main problem of KDFC lies in its software implementation. Since the feedback function is not fixed, look up tables cannot be used in the implementation of the σ-LFSR. Further, the choice of the feedback configurations is not restricted to the set of efficiently implementable σ-LFSR configurations as given in ([14]). This makes the implementation of KDFC SNOW extremely challenging.

In our implementation, the state of the σ-LFSR is stored as a set of 32 integers. The i-th integer corresponds to the i-th output of the delay blocks. Calculating the feedback function of the σ-LFSR involves calculating the bitwise XOR of a subset of the columns of the feedback matrices (B_is). In order to make the implementation more efficient, for all $1 \leq i \leq 32$, the i-th columns of the feedback matrices are stored in adjacent memory locations. Thus, each integer in the state of the σ-LFSR corresponds to a set of columns of the feedback matrices which are stored in a contiguous block of memory. The state vector is now sampled one integer at a time and the columns of the B_is corresponding to the non zero bits in these integers are XORed with each other. We then do a bit-wise right shift on each of these integers and introduce the result of the XOR operation bitwise as the most significant bits. In this way, the σ-LFSR can be implemented using bitwise XORs and shifts. The FSM is implemented as in SNOW 2.0 [8]. This implementation takes 25 cycles to generate a single word on an Intel Probook 4440s machine with a 2.8 Ghz i5 processor.

Each iteration of Algorithm 1 involves solving a system of linear equations. This process is time consuming and contributes to increasing the initialization time. The initialization process was implemented using a C code with open mp (with 3 threads). In this implementation linear equations were solved using a parallel implementation of the LU decomposition algorithm.

VI. CONCLUSION

In this paper, we have described a method of using σ-LFSRs with key-dependent feedback configurations in stream ciphers that use word-based LFSRs. In this method, an iterative configuration generation algorithm(CGA) uses key-dependant random numbers to generate a random feedback configuration for the σ-LFSR. We have theoretically analysed the algebraic degree of the resulting feedback configuration As a test case, we have demonstrated how this scheme can be used along with the Finite State Machine of SNOW 2.0. We have analysed the security of the resulting key-stream generator against various attacks and have demonstrated the improvement in security as compared to SNOW 2.0. Further, the key streams generated by the proposed method are comparable to SNOW 2.0 from a randomness point of view.

ACKNOWLEDGMENT

The authors are grateful to Prof. Harish K. Pillai, Department of Electrical Engineering, Indian Institute of Technology Bombay, for his valuable guidance and Associate Prof. Gaurav Trivedi, Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, for helping them with computational resources.

REFERENCES

[1] D. Afdhila, S. M. Nasution, and F. Azmi, "Implementation of stream cipher Salsa20 algorithm to secure voice on push to talk application," in *Proc. IEEE Asia Pacific Conf. Wireless Mobile (APWiMob)*, Sep. 2016, pp. 137–141.

[2] M. Pistono, R. Bellafqira, and G. Coatrieux, "Secure processing of stream cipher encrypted data issued from IoT: Application to a connected knee prosthesis," in *Proc. 41st Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2019, pp. 6494–6497.

[3] S. A. Jassim and A. K. Farhan, "A survey on stream ciphers for constrained environments," in *Proc. 1st Babylon Int. Conf. Inf. Technol. Sci. (BICITS)*, Apr. 2021, pp. 228–233.

[4] A. K. Farhan, "Proposed hybrid approach of stream cipher base on selector of encryption operation and key symmetric translate," *Eng. Tech.*, vol. 29, no. 11, pp. 1–11, 2011.

[5] F. Kadhim and H. Mhaibes, "Quantum random bits generator based on phase noise of laser," *J. Eng. Appl. Sci.*, vol. 13, no. 3, pp. 629–633, 2018.

[6] G. Rose and P. Hawkes. (1999). *The T-Class of Sober Stream Ciphers*. Unpublished manuscript. [Online]. Available: http://www.home.aone.net.au/qualcomm

[7] P. Ekdahl and T. Johansson, "Snow—A new stream cipher," in *Proc. 1st Open NESSIE Workshop*, Leuven, Belgium: KU-Leuven, 2000, pp. 167–168.

[8] P. Ekdahl and T. Johansson, "A new version of the stream cipher snow," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, St. John's, NL, Canada: Springer, Aug. 2002, pp. 47–61.

[9] G. G. Rose and P. Hawkes, "Turing: A fast stream cipher," in *Proc. Int. Workshop Fast Softw. Encryption*. Lund, Sweden: Springer, Feb. 2003, pp. 290–306.

[10] P. Kitsos, G. Selimis, and O. Koufopavlou, "High performance Asic implementation of the snow 3G stream cipher," in *Proc. IFIP/IEEE VLSI-SOC*, Oct. 2008, pp. 13–15.

[11] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, and T. Pornin, "SOSEMANUK, a fast software-oriented stream cipher," in *New Stream Cipher Designs*. Cham, Switzerland: Springer, 2008, pp. 98–118.

[12] F. Xiu-Tao, "ZUC algorithm: 3GPP LTE international encryption standard," *Inf. Secur. Commun. Privacy*, vol. 19, no. 12, pp. 45–46, 2011.

[13] P. Ekdahl, T. Johansson, A. Maximov, and J. Yang, "A new SNOW stream cipher called SNOW-V," *IACR Trans. Symmetric Cryptol.*, vol. 3, no. 3, pp. 1–42, 2019.

[14] G. Zeng, W. Han, and K. He, "High efficiency feedback shift register: Sigma-LFSR," *IACR Cryptol. ePrint Arch.*, vol. 2007, p. 114, Jan. 2007.

[15] S. Krishnaswamy and H. K. Pillai, "On the number of linear feedback shift registers with a special structure," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1783–1790, Mar. 2012.

[16] S. Krishnaswamy and H. K. Pillai, "On multisequences and their extensions," 2012, *arXiv:1208.4501*.

[17] O. Billet and H. Gilbert, "Resistance of snow 2.0 against algebraic attacks," in *Proc. Cryptographers' Track RSA Conf.*, San Francisco, CA, USA: Springer, Feb. 2005, pp. 19–28.

[18] D. Watanabe, A. Biryukov, and C. De Canniere, "A distinguishing attack of snow 2.0 with linear masking method," in *Proc. Int. Workshop Sel. Areas Cryptogr.* Ottawa, ON, Canada: Springer, 2003, pp. 222–233.

[19] K. Nyberg and J. Wallén, "Improved linear distinguishers for snow 2.0," in *Proc. Int. Workshop Fast Softw. Encryption*. Graz, Austria: Springer, Mar. 2006, pp. 144–162.

[20] J.-K. Lee, D. H. Lee, and S. Park, "Cryptanalysis of SOSEMANUK and snow 2.0 using linear masks," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2008, pp. 524–538.

[21] B. Zhang, C. Xu, and W. Meier, "Fast correlation attacks over extension fields, large-unit linear approximation and cryptanalysis of snow'2.0," in *Proc. Annu. Cryptol. Conf.* Cham, Switzerland: Springer, 2015, pp. 643–662.

[22] S. Kiyomoto, T. Tanaka, and K. Sakurai, "K2: A stream cipher algorithm using dynamic feedback control," in *Proc. Secrypt*, 2007, pp. 204–213.

[23] S. Ma and J. Guan, "Improved key recovery attacks on simplified version of K2 stream cipher," *Comput. J.*, vol. 64, no. 8, pp. 1253–1263, Aug. 2021.

[24] J. D. Golić, "Cryptanalysis of alleged a5 stream cipher," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1997, pp. 239–255.

[25] N. Ahmad, A. and Nanda, and K. Garg, "Critical role of primitive polynomials in an LFSR based testing technique," *Electron. Lett.*, vol. 24, no. 15, pp. 953–956, 1998.

[26] K. Chen, M. Henricksen, W. Millan, J. Fuller, L. Simpson, E. Dawson, H. J. Lee, and S. J. Moon, "Dragon: A fast word based stream cipher," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Seoul South Korea, Dec. 2004, pp. 33–50.

[27] S. Krishnaswamy and H. K. Pillai, "On the number of special feedback configurations in linear modular systems," *Syst. Control Lett.*, vol. 66, pp. 28–34, Apr. 2014.

[28] H. Ahmadi and T. Eghlidos, "Heuristic guess-and-determine attacks on stream ciphers," *IET Inf. Secur.*, vol. 3, no. 2, pp. 66–73, Jun. 2009.

[29] M. Hell, T. Johansson, and L. Brynielsson, "An overview of distinguishing attacks on stream ciphers," *Cryptogr. Commun.*, vol. 1, no. 1, pp. 71–94, Apr. 2009.

[30] B. Debraize and I. M. Corbella, "Fault analysis of the stream cipher snow 3G," in *Proc. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC)*, Sep. 2009, pp. 105–112.

[31] M. S. N. Nia and A. Payandeh, "The new heuristic guess and determine attack on snow 2.0 stream cipher," *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 619, Jan. 2014.

[32] N. T. Courtois and B. Debraize, "Algebraic description and simultaneous linear approximations of addition in snow 2.0," in *Proc. Int. Conf. Inf. Commun. Secur.* Springer, 2008, pp. 328–344.

[33] M. Matsui, "Linear cryptanalysis method for des cipher," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Springer, 1993, pp. 386–397.

[34] W. Meier and O. Staffelbach, "Fast correlation attacks on certain stream ciphers," *J. Cryptol.*, vol. 1, no. 3, pp. 159–176, 1989.

[35] V. V. Chepyzhov, T. Johansson, and B. Smeets, "A simple algorithm for fast correlation attacks on stream ciphers," in *Proc. Int. Workshop Fast Softw. Encryption*. Berlin, Germany: Springer, 2000, pp. 181–195.

[36] F. Jonsson and T. Johansson, "Correlation attacks on stream ciphers over GF(2/sup n/)," in *Proc. IEEE Int. Symp. Inf. Theory*, May 2001, p. 140.

[37] D. Wagner, "A generalized birthday problem," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2002, pp. 288–304.

[38] Y. Todo, T. Isobe, W. Meier, K. Aoki, and B. Zhang, "Fast correlation attack revisited," in *Proc. Annu. Int. Cryptol. Conf.* Springer, 2018, pp. 129–159.

[39] X. Gong and B. Zhang, "Fast computation of linear approximation over certain composition functions and applications to snow 2.0 and snow 3G," *Des., Codes Cryptogr.*, vol. 88, no. 11, pp. 2407–2431, 2020.

[40] G. Leander, E. Zenner, and P. Hawkes, "Cache timing analysis of LFSR-based stream ciphers," in *Proc. IMA Int. Conf. Cryptogr. Coding* Cham, Switzerland: Springer, 2009, pp. 433–445.

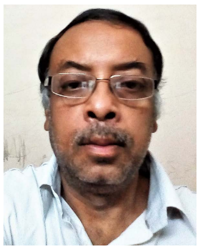
[41] B. B. Brumley, R. M. Hakala, K. Nyberg, and S. Sovio, "Consecutive s-box lookups: A timing attack on snow 3G," in *Proc. Int. Conf. Inf. Commun. Secur.* Norwell, MA, USA: Springer, 2010, pp. 171–185.

[42] E. Yılmaz, "Two versions of the stream cipher snow," M.S. thesis, Dept. Elect. Electron. Eng., Middle East Tech. Univ., Ankara, Turkey, 2004.

SUBRATA NANDI received the M.Tech. degree in computer science engineering from KIIT University, Bhubaneswar, India. He worked as a Project Fellow with the Mathematics Department, National Institute of Science Education and Research (NISER), Bhubaneswar, India, in a DST Funded Project. He is currently working as a Ph.D. Research Scholar in Cryptology at the CSE Department, Indian Institute of Technology (IIT) Guwahati, under the supervision of Dr. Srinivasan Krishnaswamy and Dr. Pinkai Mitra. His research interests include $\sigma-$LFSR, word-based LFSR, and cryptographic Boolean function.

SRINIVASAN KRISHNASWAMY received the Ph.D. degree in electrical engineering from the Indian Institute of Technology Bombay, Mumbai. He is currently an Assistant Professor at the Indian Institute of Technology (IIT) Guwahati, India. His research interests include cryptography and control systems.

BEHROUZ ZOLFAGHARI received the Ph.D. degree in computer engineering from the Amirkabir University of Technology, Tehran, Iran. He worked as a Postdoctoral Research Fellow at the Indian Institute of Technology (IIT) Guwahati, India. His research interests include information theory, information-theoretic cryptography, hardware oriented cryptography, and AI-assisted cryptography.



PINAKI MITRA received the B.Tech. degree in computer science and engineering from Jadavpur University, Kolkata, India, in 1987, the M.Tech. degree in computer science and engineering from the Indian Institute of Science Bengaluru, India, in 1989, and the Ph.D. degree from Simon Fraser University, Canada, in 1994. He worked on a project at the Department of Computer Science and Engineering, Jadavpur University. Subsequently, he joined the National Institute of Management, Kolkata, and worked as an Assistant Professor. He joined the IIT Guwahati, in December 2004. He is currently an Associate Professor at the Department of Computer Science and Engineering, IIT Guwahati. His research interests include cryptography, network security, computer graphics, and multimedia.