Key-Dependent Feedback Configuration matrix of primitive $\sigma - LFSR$ and resistance to some Known Plaintext Attacks

SUBRATA NANDI$^1$, SRINIVASAN KRISHNASWAMY$^2$, BEHROUZ ZOLFAGHARI$^3$ AND PINAKI MITRA$^4$

$^1$CSE Department, Indian Institute of Technology Guwahati, Gwa 781039, India (email: subrata.nandi@iitg.ac.in)
$^2$EEE Department, Indian Institute of Technology Guwahati, Gwa 781039, India (email: srinikris@iitg.ac.in)
$^3$CSE Department, Indian Institute of Technology Guwahati, Gwa 781039, India (email: zolfaghari@iitg.ac.in)
$^4$CSE Department, Indian Institute of Technology Guwahati, Gwa 781039, India (email: pinaki@iitg.ac.in)

Corresponding author: First A. Author (e-mail: subrata.nandi@iitg.ac.in, ORCID ID: 000-0001-7140-0608)

ABSTRACT In this paper, we propose and evaluate a method for generating key-dependent feedback configurations (KDFC) for $\sigma - LFSR$s. $\sigma - LFSR$s 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 attack. Besides, as a consequence of resisting algebraic attack, 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 $\sigma - LFSR$ 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, 7, 8, 9, 10, 11, 12, 13]). Such LFSRs are used in a variety of stream ciphers, most notably in the SNOW series of stream ciphers. A $\sigma - LFSR$ is a word based LFSR configuration that was introduced in [14]. An important property of this configuration is that there are multiple feedback functions 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 LFSR based stream ciphers. These include algebraic attacks, correlation attacks, distinguishing attacks, guess and determining attack and Cache timing attack ([17, 18, 19, 20, 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 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 $\sigma$-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 nonlinear 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 $\sigma$-KDFC with the finite state machine (FSM) of 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.

### A. SYMBOL TABLE

| Symbol | Meaning |
|--------|---------|
| $F_{2^n}$ | Finite field of cardinality $2^n$ |
| $F^n_2$ | $n$-dimensional vector space over $F_2$ |
| $M_{i,j}$ | The $i^{th}$ row of a matrix $M$ |
| $M_{i,j}$ | $j^{th}$ column of a matrix $M$ |
| $\mu(M_{i,j})$ | The minor of $M_{i,j}$ |
| $+$ | Addition or XOR |
| $M_{n \times n}$ | Matrix $M$ with $n$ rows and $n$ columns |
| $e_{i}^{T}$ | Vector with $n$-th coordinate is 1 and rest are 0. |
| $M^{-1}$ | Inverse of a matrix $M$. |

### TABLE 1. Symbol and their meaning

The rest of this paper is organized as follows. Section II-A introduces LFSRs, $\sigma$-LFSRs and some related concepts. Section III examines $\sigma$-KDFC and its time complexity. Section IV presents the mathematical analysis on the algebraic degree of the parameters of the feedback function generated by $\sigma$-KDFC. Section V discusses the the interconnection of $\sigma$-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 $\sigma$-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 that time instant i.e $x_n = [x_n, x_{n+1}, \ldots, x_{n+b-1}]$. Two consecutive state vectors are related by the equation $x_{n+1} = x_nP_f$ where $P_f$ is the companion matrix of the characteristic polynomial.

\[
P_f = \begin{bmatrix}
0 & 0 & \cdots & 0 & a_0 \\
1 & 0 & \cdots & 0 & a_1 \\
0 & 1 & \cdots & 0 & a_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & a_{b-1}
\end{bmatrix}
\] (1)

In order to efficiently work with word based processors various word based LFSR designs have been proposed [6, 8, 11, 13, 26, 27, 28]. These designs use multi input multi output delay blocks. One such design is the $\sigma$-LFSR shown in Figure 1. Here, the feedback gains are matrices and the implemented linear recurring relation is of the form

\[
x_{n+b} = B_{n+b-1}x_{n+b-1} + B_{n+b-2}x_{n+b-2} + \cdots + B_0x_n
\] (2)

where each $x_i \in F_2^n$ and $B_i \in F_2^{m \times m}$. Here, each delay block has $m$-inputs and $m$-outputs and the $\sigma$-LFSR generates a sequence of vectors in $F_2^m$.

The matrices $B_0, B_1, \ldots, B_{b-1}$ are referred to as the gain matrices of the $\sigma$-LFSR and the following matrix is defined as its configuration matrix.
where 0, I ∈ F_m^{m×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 \( \tilde{x}_n \) is the output of the σ-LFSR at the \( n \)-th time instant, then its state vector at that time instant is defined as \( \tilde{x}_n = [x_n, x_{n+1}, \ldots, x_{n+b-1}] \). 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:

\[
\tilde{x}_{n+1} = \tilde{x}_n C^T
\] (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,

\[
N_P = \frac{|GL(m, F_2)|}{2^m - 1} \times \phi(2^mb - 1) \times 2^{m(m-1)(b-1)}
\] (5)

where 5, \( GL(m, F_2) \) is the general linear group of non-singular matrices ∈ \( F_2^{m×m} \), and \( \phi \) represents Euler’s totient. This conjecture has been proved in [29]. 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 [29] 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, 30]. 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 [29] 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 \( F_2^{m \times m} \).
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 \( mb - m \)
5. if \( t \neq \ell \) mod \( m \) then
6. end if
7. end for
9. \( \ell \) \{At the end of each iteration, an extra column is added to \( Y \) till \( Y \in F_2^{mb \times mb} \).
11. Construct the following matrix \( Q \). 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 \( d_i \)s 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 \( y \times K_i = (0, 0, \ldots, 1) \in F_2^{m+i-1} \) for \( y \), where \( K_i \) is given by

\[
C = \begin{bmatrix}
0 & I & 0 & \cdots & 0 \\
0 & 0 & I & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & I \\
B_0 & B_1 & B_2 & \cdots & B_{b-1}
\end{bmatrix} \in F_2^{mb \times mb} \]
In other words, $f(x) = y(1) + y(2)x + \cdots + y(m + i - 1)x^m + \cdots + y(m + i - 2)$, 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 III.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 $x^8 + x^4 + x^3 + x^2 + 1$ which is a primitive polynomial of degree 8.

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

$$
\begin{pmatrix}
1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1
\end{pmatrix}_{4 \times 4} \xrightarrow{xM^i} \begin{pmatrix}
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1
\end{pmatrix}_{4 \times 5}
$$

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

This results in the following matrix $C$.

$$
\begin{pmatrix}
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0
\end{pmatrix}_{4 \times 6}
$$

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

$$
\begin{pmatrix}
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0
\end{pmatrix}_{4 \times 5} \xrightarrow{xM^i} \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1
\end{pmatrix}_{4 \times 6}
$$

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

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1
\end{pmatrix}_{4 \times 5} \xrightarrow{xM^i} \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}_{4 \times 4}
$$

The corresponding matrix $Q$ is as follows.

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

The following matrix $P$ is the companion matrix of the primitive polynomial $x^5 + x^4 + x^3 + x^2 + 1$.

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

This results in the following matrix $C$. 

FIGURE 2. The Schematic of $\sigma$--KDFC

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
The computational capacity of the machine that hosts the matrix \( Y \) is known feedback configuration. Further, the pre-calculated detail.

During the initialization phase the \( \sigma \)-LFSR configuration is derived from the FSM. The feedback configuration 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 \( \sigma \)-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 \( \sigma \)-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 \( p_m+k(x), p_{m+k+2}(x), \ldots, p_{mb}(x) \) are also publicly known. The initial state of the \( \sigma \)-LFSR is derived from the secret key and the IV. (as is normally done in word based stream ciphers like SNOW). The \( \sigma \)-LFSR is run along with the FSM for \( mb - m - k \) clock cycles. This generates \( mb - m - k \) vectors in \( \mathbb{F}_2^{n} \). 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 \( \sigma \)-LFSR are now set according to the configuration matrix that is generated by Algorithm 1. Once the feedback gains are set the \( \sigma \)-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 \( \sigma \)-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 \( O((mb)^3) \) 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 \( O(1) \), then the overall time complexity of Step 3 is \( O((mb)^3) \). 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 \( O((mb)^4) \) using Gaussian elimination. Step 4 has a time complexity of \( O((mb)^3) \). Thus, the time complexity of the initialization phase is \( O((mb)^4) \).

### IV. ALGEBRAIC ANALYSIS OF \( \sigma \)-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 \( \sigma \)-LFSR.

Note that the last row of \( Y \) is always \( e_1^{m} \). 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,

\[
B_k(i,j) = f_k(i,j)(\mathcal{U}) \quad \text{for } 0 \leq k \leq b-1 \text{ and } 1 \leq i, j \leq m
\]

where \( f_k(i,j) \) are polynomial functions.

The algebraic degree of the configuration matrix, denoted by \( \Theta \), is defined as follows

\[
\Theta(C_S) = \max_{k,i,j} |f_k(i,j)(\mathcal{U})|
\]

\( \Theta \) can be considered as a measure of the algebraic resistance of \( \sigma \)-KDFC. We now proceed to find a lower bound for \( \Theta \).

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

\[
Q = \begin{bmatrix}
  v_1 \\
v_2 \\
  \vdots \\
v_{m-1} \\
e_1^P mb \\
  v_1 P_{mb} \\
  \vdots \\
v_1 P_{mb}^{b-1} \\
v_{m-1} P_{mb}^{b-1} \\
v_{m-1} P_{mb}^{b-1} \\
\end{bmatrix}
\]  

(8)

where \( P_{mb} \) is the companion matrix of the publicly known primitive characteristic polynomial of the \( \sigma \)-LFSR. The configuration matrix \( C \) is generated by the formula \( C = Q \times P_{mb} \times Q^{-1} \). Since \( Q \) is an invertible 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 \( Q_P \)

\[
Q_P = \begin{bmatrix}
  e_1^n \\
e_1^P mb \\
  \vdots \\
  e_1 P_{mb}^{b-1} \\
e_1 P_{mb}^{b-1} \\
  \vdots \\
v_1 P_{mb}^{b-1} \\
v_{m-1} P_{mb}^{b-1} \\
v_{m-1} P_{mb}^{b-1} \\
v_{m-1} P_{mb}^{b-1} \\
\end{bmatrix}
\]  

(9)

where \( Q_P \) can be decomposed as follows into four sub-matrices \( Q_1, Q_2, Q_3 \) and \( Q_4 \):

\[
Q_P = \begin{bmatrix}
  Q_1 & Q_2 \\
  Q_3 & Q_4
\end{bmatrix}
\]  

(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

\[
Q_2 = \begin{bmatrix}
  0 & \cdots & 0 \\
  0 & \cdots & 1 & * \\
  \vdots & \ddots & \vdots \\
  1 & \cdots & * & *
\end{bmatrix}
\]  

(11)

\[
Q_3 = \begin{bmatrix}
  v_{1,1} & v_{1,2} & \cdots & v_{1,mb-b} \\
  v_{1,2} & v_{1,3} & \cdots & v_{1,mb-b+1} \\
  \vdots & \vdots & \ddots & \vdots \\
  v_{m-1,1} & v_{m-1,2} & \cdots & v_{m-1,mb-b}
\end{bmatrix}
\]  

(12)

\[
Q_4 = \begin{bmatrix}
  v_{1,mb-b+1} & \cdots & v_{1,mb-1} & v_{1,mb} \\
  v_{1,mb-b+2} & \cdots & v_{1,mb} & * \\
  \vdots & \ddots & \vdots \\
  v_{2,mb-1} & \cdots & \cdots & * \\
  v_{m-1,mb-b+1} & \cdots & v_{m-1,mb-1} & v_{m-1,mb}
\end{bmatrix}
\]  

(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 \( Q_P \). Since \( Q_P \) is invertible, \( \text{det}(Q_P) = \text{det}(Q_3) = 1 \).

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

**Lemma IV.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 \( [AB]_{[0],a} \) 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:

\[
\mu(Q_P[i,j]) = \text{det}([Q_3Q_4]_{ij,1})
\]  

(14)
Recall that, for a binary matrix $M \in \mathbb{F}_{2}^{mb \times mb}$, its determinant is given by the following formula,
\[
det(M) = \sum_{f \in S_{mb}} \prod_{1 \leq i \leq n} M(i, f(i))
\]
(15)
where $S_{mb}$ is the set of permutations on $(1, 2, \ldots, mb)$. Observe that the diagonal elements of $[(Q_{3}Q_{4})]_{j,1}$ are distinct $v_{i,b}$. Their product corresponds to the identity permutation in the determinant expansion formula for $[Q_{3}Q_{4}]_{j,1}$. The resultant monomial has degree $mb-b$. Further, this monomial will not occur as a result of any other permutation. Hence $\det([Q_{3}Q_{4}]_{j,1})$ is always a polynomial of degree $mb-b$. \hfill $\square$

Lemma IV.2. If $1 \leq i \leq b$ then
\[
\mu(Q_{P}[i,j]) = \left\{ \begin{array}{ll}
det(Q_{3}) & i + j = mb + 1 \\
0 & i + j > mb + 1
\end{array} \right.
\]
(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 $Q_{2}$ 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, $\mu(Q_{P}[i,j]) = 0$. \hfill $\square$

Lemma IV.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 $v_{i,j}$. 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, $\mu(Q_{P}[i,j]) \in \Gamma_{mb-b-1}$. \hfill $\square$

Lemma IV.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, $\mu(Q_{P}[i,j]) = 0$. \hfill $\square$

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 IV.1.
\[
\Theta(Q_{P}^{-1}) = \Theta(Q_{P}^{-1}) = mb - b
\]
(17)

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

Theorem IV.5. $\Theta(C) \geq mb - b$.

Proof. Observe that the gain matrices $B_{0}, B_{1}, \ldots, B_{b-1}$ appear in the last $m$ rows of $C_{S}$. 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
\[
\begin{pmatrix}
0 & 0 & \cdots & 1 & \cdots & \ast & \ast & \ast
\end{pmatrix}
\]
\[
\begin{pmatrix}
v_{1,b+1} & v_{1,b+2} & \cdots & v_{1,mb} & \ast & \ast & \ast & \ast
\end{pmatrix}
\]
\[
\begin{pmatrix}
v_{2,b+1} & v_{2,b+2} & \cdots & v_{2,mb} & \ast & \ast & \ast & \ast
\end{pmatrix}
\]
\[
\vdots
\end{pmatrix}
\]
\[
\begin{pmatrix}
v_{m-1,b+1} & v_{m-1,b+2} & \cdots & v_{m-1,mb} & \ast & \ast & \ast & \ast
\end{pmatrix}
\]
(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 $Q^{-1}$. As a consequence of Lemmas IV.1 and IV.2, this column has the following form.
\[
Q^{-1}[\cdot, mb - m + 1] = (P_{1}, P_{2}, \cdots, P_{mb-1}, 1, 0, \cdots, 0)^{T}
\]
(19)
where $P_{1}, P_{2}, \cdots, P_{mb-1} \in \Gamma(mb - b)$. Therefore,
\[
C[mb - b + 1, mb - b + 1] = \left( (mb - b) \text{ entries} \right)
\times (P_{1}, P_{2}, \ldots, P_{mb-1}, 1, 0, \cdots, 0)^{T}
\]
\[
= P_{mb-b}
\]

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

Example IV.1. 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 $P_{2}$ is $f(x) = x^{8} + x^{4} + x^{3} + x^{2} + 1$. The corresponding matrix $Q_{P}$ has the following submatrices.

\[
Q_{1} = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]
(20)

\[
Q_{2} = \begin{pmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0
\end{pmatrix}
\]
(21)
attacks. We consider SNOW 1.0. This version was shown to be vulnerable to algebraic and was introduced later in [8] as a modified version of SNOW 0. This version of SNOW is known as KDFC-SNOW.

\[ \sigma \] is the root of the following primitive polynomial

\[ x^{256} + x^{224} + x^{192} + x^{128} + 1 \in \mathbb{F}_2[X] \]

where \( \beta \) 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 \( R_1 \) and \( R_2 \). 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 \( \sigma \)-LFSR having 16, 32-input 32-output delay blocks. The configuration matrix of the \( \sigma \)-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 \( \sigma \)-LFSR is identical to that of SNOW-2. As in SNOW 2.0, the \( \sigma \)-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

\[ Q_3 = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \\ x_4 & x_5 & x_6 & x_7 \end{pmatrix} \]

(22)

Therefore,

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

where

\[ P_1 = x_2x_4x_6x_8 + x_2x_4x_7 + x_2x_5x_8 + x_2x_6 + x_3x_6x_8 + x_3x_7 + x_4x_6 + x_4x_7 + x_5 \]

\[ P_2 = x_1x_4x_6x_8 + x_1x_4x_7 + x_1x_5x_8 + x_1x_6 + x_2x_3x_6x_8 + x_2x_3x_7 + x_2x_4x_5x_8 + x_2x_4x_6x_7 + x_2x_5x_6 + x_2x_5x_7 + x_3x_4x_5 + x_3x_5x_6 + x_3x_5x_7 + x_3x_6 + x_4x_5 + x_4x_7 \]

\[ P_3 = x_1x_3x_5x_3 + x_1x_3x_4 + x_1x_5x_8 + x_1x_5x_9 + x_1x_5x_10 + x_2x_3x_6x_8 + x_2x_3x_6x_7 + x_2x_4x_5x_8 + x_2x_4x_6x_7 + x_3x_4x_5 + x_3x_5x_6 + x_3x_5x_7 + x_3x_6 + x_4x_5 + x_4x_6 + x_4x_7 \]

\[ P_4 = (x_1x_3x_5x_8 + x_1x_3x_6x_7 + x_1x_4x_5 + x_1x_5x_6 + x_2x_3x_5x_4 + x_2x_3x_6x_7 + x_2x_4x_5x_8 + x_2x_4x_6x_7 + x_3x_4x_5 + x_3x_5x_6 + x_3x_5x_7 + x_3x_6 + x_4x_5 + x_4x_7 + x_4x_6 + x_4x_7 + x_5 + x_6) \]

Here, \( C_S[7, 7] \), 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 \( \sigma \)-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 [28]. 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 [31].

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, 30, 31, 32, 33]. We consider SNOW 2.0 as a test case and demonstrate how replacing the LFSR in this scheme with a \( \sigma \)-LFSR increases its resistance to various attacks.

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

\[ \begin{pmatrix} x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} x_8 \\ x_9 \\ x_{10} \end{pmatrix} \]

\[ \begin{pmatrix} x_1 + x_3 + x_4 + x_5 \\ x_2 + x_4 + x_5 + x_6 \end{pmatrix} = \begin{pmatrix} x_7 + x_9 + x_{10} \\ x_8 + x_9 + x_{10} \end{pmatrix} \]

(23)

\[ x_{n+16} = \alpha^{-1}x_{n+11} + x_{n+2} + \alpha x_0 \]

Therefore,

\[ Q_P^{-1}[4] = [Q^{-1}[4], 7] = (P_1, P_2, P_3, P_4, 1, 0, 0, 0) \]
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 $D_0, \ldots, D_{15}$ 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 $\delta$-LFSR is identical to SNOW 2.0. This is done by setting $B_{13}$ and $B_0$ as matrices that represent multiplication by $\alpha^{-1}$ and $\alpha$, 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 $33^{rd}$ vector.

**D. GOVERNING EQUATIONS OF KDFC-SNOW**

Let $D^t_i \in \mathbb{F}_{2^2}$ 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 $\delta$-LFSR are related as per the following equation:

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

(24)

Therefore,

$$D_{k}^{t+1} = \begin{cases} D_0^{k+1} & 0 \leq k + t + 1 \leq 15 \\ \sum_{i=0}^{15} B_i D_i^{t+k} & k + t + 1 > 15 \end{cases}$$

(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 $t$,

$$F_t = z^t + D_0^t = (D_{15}^{t+1} \boxplus R_1^t) + R_2^t$$

(26)

The registers are updated as follows:

$$R_1^{t+1} = D_0^t \boxplus R_2^t$$

(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)

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:

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

(30)

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

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 this 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

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

(31)

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

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

(32)

Further, Equation 28 can be expanded as follows:

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

(33)

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

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

(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 $\sum_{i=0}^{2^{17}} (\binom{544}{i}) \approx 2^{17}$. Therefore we have a system of equations, which can be solved in $O(2^{51})$ 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 $O(2^{294})$. 

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3141434, IEEE Access
When the LFSR in SNOW 2.0 is replaced by a \( \sigma \)-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 \( 16 \times m^2 + mb + m = 16928 \) unknown variables (This includes the \( 16 \times m^2 \) 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 \( R_2^t \) and \( R_2^{t+1} \), \( D_1^t \) and \( D_1^{t+1} \) are given as follows:

\[
D_1^t = B_0 D_0^t + B_1 D_1^0 + \ldots + B_{15} D_{15}^t \\
D_1^{t+1} = B_0 D_0^{t+1} + B_1 D_1^0 + \ldots + B_{15} D_{15}^{t+1}
\]

Observe that, while \( D_1^t \) is a polynomial of degree two in the unknown variables, \( D_1^{t+1} \) is a polynomial of degree 3. Similarly, with each successive iteration the degree of the expression for \( D_1^{t+1} \) keeps increasing, till all the \( m^2b \) entries of the feedback matrices are multiplied with each other. This 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 \( \sigma \)-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 \( N = \sum_{i=0}^{M} (16928) \) 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 \( e_1 \), the total number of unknowns will now be \( 31 \times 512 = 15872 \). As we have already seen, the entries of the feedback matrices \( (B_i) \) are polynomials in these variables. From Theorem 4.5, the maximum degree of these polynomials is at least \( mb - b \). Therefore, the maximum degree of the equations generated by Equation 33 will be at least \( mb - b + 1 = 497 \). Therefore, linearizing this system of equations gives rise to a system of linear equations in \( N = \sum_{i=0}^{497} (16416) \approx O(2^{3207}) \) 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 [34]. 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 feedback matrices and their products. For example, \( B_0 \) and \( B_1 \) are unknowns. Assuming that the feedback matrices are considered as unknowns, then the value of \( R_1 \) uniquely determines the value of \( R_2^{t+1} \) (by Equation 28). Given values of \( R_1 \) and \( R_2 \), Equation 26 gives rise to two linear equations and thirty quadratic equations. Since the relation between the current state and the initial state is linear, one can get an estimate of the initial state from the equation obtained from the feedback equation of the LFSR and the register \( R_2 \) at the beginning of the key stream. This accounts for a total of 224 equations. Further, there are 7 consecutive values of \( R_1 \) and \( R_2 \) are simultaneously known. Hence, at each of these time instances, Equation 27 gives rise to 32 linear equations over \( GF(2) \). 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 register \( R_2 \) is given as follows:

\[
R_1^t = 0, R_1^{t+1} = 2^{32} - 1, R_2^{t+2} = 0, \ldots, R_2^{t+8} = 0 \tag{35}
\]

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 \( GF(2) \) 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 register \( R_2 \) 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 [34], this results in Equation 26 generating 32 linear equations which are satisfied with probability half. This probability becomes 1 when \( D_5^{t+1} + 2^2 + S(0) \) is zero. We thus have a total of 512 linear equations. 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 significantly 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, 19, 20]. This method essentially adapts the linear cryptanalysis method given in [35] 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:

\[ \sum_{i=0}^{\ell_1} \Gamma_i x_{n+i} = \sum_{i=0}^{\ell_2} \Lambda_i x_{n+i} \quad (37) \]

where \( z_0, z_1, \ldots \) is the output sequence of the key stream generator. The \( \Gamma_i \)s and \( \Lambda_i \)s are linear masks that map the corresponding \( x_{n+i} \)s and \( z_{n+i} \)s to \( F_2 \) 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 [35]. The main task in this type of attack is to find masks \( \Gamma_i \)s and \( \Gamma'_i \)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.

\[
\begin{align*}
\Gamma_0 x_n &+ \Gamma_1 x_{n+1} + \Gamma_5 x_{n+5} + \Gamma_{15} x_{n+15} + \Gamma_{16} x_{n+16} = \\
&= \Lambda_0 z_n + \Lambda_1 z_{n+1}
\end{align*}
\]

In [18] it is assumed that all the \( \Gamma_i \)s and \( \Lambda_i \)s are equal to each other. In [19] it is assumed that \( \Gamma_0, \Gamma_{15} \) and \( \Lambda_0 \) are equal to each other. \( \Gamma_1, \Gamma_5, \Gamma_{16} \) and \( \Lambda_1 \) 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 \( z_i \)s:

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

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

In the proposed \( \sigma \)-LFSR configuration, the feedback equation is not known. Therefore the only known linear recurring relation that the output of the \( \sigma \)-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 \( F_2 \), 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 \( \epsilon \), then the bias of the relation between the elements of the key stream is given as follows

\[
\epsilon_{\text{final}} = 2^{249} \times \epsilon^{250} \quad (38)
\]

The number of elements of the key stream needed to distinguish it from a random sequence is \( \frac{1}{\epsilon} \). 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.

| Reference | \( \epsilon \) | \( \epsilon_{\text{final}} \) SNOW 2.0 | \( \epsilon_{\text{final}} \) KDFC SNOW | \#Keystream SNOW 2.0 | \#Keystream KDFC SNOW |
|-----------|---------------|----------------|----------------|------------------|------------------|
| [18]      | 2 \times 2^{112.25} | 2 \times 2^{5623.66} | 2^{224} | 2^{3623} | 2^{213} |
| [19]      | 2 \times 2^{114.49} | 2 \times 2^{366.9} | 2^{225} | 2^{3623} | 2^{214} |

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

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 (36). 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 [37]. 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 [38]. An improvement on this attack is given in [20]. Both these schemes consider a linear recurring relation with coefficients in \( F_2 \). 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 $2^{12.38}$. 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 [39] 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 $\sigma$-LFSR is publicly known. The attacker can therefore generate a linear recurring relation over $\mathbb{F}_{2}$ that the output of the $\sigma$-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].

[40] 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 $2^{1.50}$ times better than [21]. [41] 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 $\mathbb{F}_{2^2}$. Hence, if the size of the basis is $k$, then the probability of a correct guess is $2^{-32k}$. Thus, on average, one needs $O(2^{-32k})$ attempts to make a correct guess. Therefore, the problem here is to find a basis of the minimum possible size. A systematic Viterbi-like algorithm for doing this is given in [30]. The complexity of this attack was found to be $2^{656}(30)$ for SNOW2.0. The complexity of this attack reduced to $2^{192}$ in ([33]) by incorporating a couple of auxiliary equations. We now briefly describe the algorithm given in [30] in the context of SNOW 2.0.

Consider the following equations which are satisfied by

$$
D_i^{16} = \alpha^{-1} D_i^{11} + D_i^{12} + \alpha D_i^{13} \tag{39}
$$

$$
R_{1i} = D_i^4 + S(R_{1i-2}) \tag{40}
$$

$$
z_t = D_t + (D_t^{15} + R_{1i}) + S(R_{1i-1}) \tag{41}
$$

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 $R_1$. 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 $\sigma$-LFSR is not known. The smallest known linear recurring relation that the output of the $\sigma$-LFSR satisfies is the relation corresponding to its characteristic polynomial. This relation is given as follows

| $x_n+512$ | $x_n+510$ | $x_n+504$ | $x_n+502$ | $x_n+501$ | $x_n+494$ | $x_n+493$ | $x_n+490$ | $x_n+486$ | $x_n+485$ | $x_n+483$ | $x_n+481$ | $x_n+480$ | $x_n+478$ | $x_n+477$ | $x_n+471$ | $x_n+470$ | $x_n+469$ | $x_n+466$ | $x_n+462$ | $x_n+461$ | $x_n+459$ | $x_n+458$ | $x_n+452$ | $x_n+449$ | $x_n+446$ | $x_n+445$ | $x_n+444$ | $x_n+441$ | $x_n+438$ | $x_n+437$ | $x_n+434$ | $x_n+433$ | $x_n+432$ | $x_n+431$ | $x_n+429$ | $x_n+427$ | $x_n+424$ | $x_n+423$ | $x_n+420$ | $x_n+419$ | $x_n+414$ | $x_n+412$ | $x_n+411$ | $x_n+409$ | $x_n+405$ | $x_n+402$ | $x_n+400$ | $x_n+399$ | $x_n+398$ | $x_n+396$ | $x_n+395$ | $x_n+393$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 11 | 16 | 4 | 35 | 37 | 0 | 15 | 36 | 38 | 18 | 20 | 29 | 34 | 4 | 35 | 37 | 0 | 15 | 36 | 38 | 18 | 23 | 34 | 55 |
| 1 | 3 | 12 | 17 | 5 | 36 | 38 | 1 | 16 | 37 | 38 | 22 | 53 | 55 | 18 | 33 | 54 | 55 |

**TABLE 3.** Index table for SNOW 2.0

SNOW 2.0

| $D_i^{16}$ | $\alpha^{-1} D_i^{11} + D_i^{12} + \alpha D_i^{13}$ \tag{39} 
| $R_{1i}$ | $D_i^4 + S(R_{1i-2})$ \tag{40} 
| $z_t$ | $D_t + (D_t^{15} + R_{1i}) + S(R_{1i-1})$ \tag{41} 

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3141434, IEEE Access
$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+234} + x_{n+233} + x_{n+230} + x_{n+229} + x_{n+228} + x_{n+227} + x_{n+226} + x_{n+222} + x_{n+221} + 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}

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

\[
R1_{n} = x_{n+4} + S(R1_{n-1}) \\
S_{n} = x_{n} + (x_{n+15} + R1_{n}) + S(R1_{n-1})
\]

The following tables can be constructed using these three equations. We ran the Viterbi-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 $O(2^{544})$.

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 [42, 43], 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 $\mathbb{F}_{2^{16}}$ that are done in every cycle of the keystream generation by $\alpha$ and $\alpha^{-1}$. These multiplications are implemented as follows

\[
\alpha \ast x = (x << 8) \oplus T_1[x_3] \quad (42)
\]
\[
\alpha^{-1} \ast x = (x >> 8) \oplus T_2[x_0] \quad (43)
\]

where $x = (x_3x_2x_1x_0) \in \mathbb{F}_{2^{16}}$ and $T_1, T_2$ are two 8 $\times$ 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 $S_{t+1}$. 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 characteristic polynomial of the $\sigma$-LFSR has been taken as $f(x) = x^{512} + x^{510} + 504 + x^{502} + 501 + x^{494} + 493 + x^{490} + 486 + 485 + 483 + 481 + x^{478} + 477 + x^{471} + 470 + 469 + x^{466} + 462 + x^{461} + 459 + 458 + x^{452} + 449 + x^{446} + 445 + 444 + 441 + x^{438} + 437 + x^{434} + 433 + x^{432} + 431 + x^{429} + 427 + 424 + x^{423} + 420 + x^{419} + 410 + x^{411} + 409 + 405 + 402 + 400 + 399 + 398 + x^{396} + 395 + x^{393} + 392 + x^{390} + 388 + x^{387} + 385 + 375 + x^{374} + 372 + x^{371} + 366 + x^{365} + 363 + x^{362} + 359 + 357 + x^{356} + 355 + 354 + x^{353} + 352 + 351 + x^{350} + 347 + x^{345} + 344 + x^{341} + 339 + x^{338} + x^{337} + x^{336} + x^{333} + x^{330} + x^{329} + x^{326} + x^{324} + x^{322} + x^{319} + 310 + x^{307} + x^{306} + x^{305} + 303 + 301 + 299 + 298 + 297 + x^{296} + x^{295} + x^{294} + 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^{154} + x^{152} + x^{151} + x^{149} + x^{147} + x^{146} + x^{143} + x^{141} + x^{138} + x^{139} + 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^{29} + x^{28} + x^{25} + x^{21} + x^{19} + x^{16} + x^{5} + 1$.

(This polynomial is the characteristic polynomial of the LFSR in SOW 2.0 when it is implemented as a $\sigma$-LFSR i.e. when multiplication by $\alpha$ and $\alpha^{-1}$ 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.

| S.No. | Test | P-Value | Random |
|-------|------|---------|--------|
| 01    | Frequency Test (Monobit) | 0.35966689490586125 | ✓ |
| 02    | Frequency Test within a Block | 0.24371484001729746 | ✓ |
| 03    | Run Test | 0.903818434231019 | ✓ |
| 04    | Longest Run of Ones in a Block | 0.5246846287441829 | ✓ |
| 05    | Binary Matrix Rank Test | 0.1377116998339736 | ✓ |
| 06    | Discrete Transform (Spectral) Test | 0.1377116998339736 | ✓ |
| 07    | Non-Overlapping Template Matching Test | 0.318981228843801 | ✓ |
| 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    | Cumulative Sums (Forward) Test | 0.34630795948969923 | ✓ |
| 14    | Cumulative Sums (Reversed) Test | 0.6633666090204551 | ✓ |
| 15    | Random Excursions Test | 0.969753280059932 | ✓ |
| 16    | Random Excursions Variant Test | 0.97387 | ✓ |

These results are comparable to that of SOW 2.0([44]).

Note that the feedback configuration of SOW 2.0 is one of the possible feedback configurations in the $\sigma$-KDFC scheme.

G. CHALLENGES IN IMPLEMENTATION

The feedback function of a $\sigma$-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 $\sigma$-LFSR. Further, the choice of the feedback configurations is not restricted to the set of efficiently implementable $\sigma$-LFSR configurations as given in ([14]). This makes the implementation of KDFC SNOW extremely challenging.

In our implementation, the state of the $\sigma$-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 $\sigma$-LFSR involves calculating the bitwise XOR of a subset of the columns of the feedback matrices ($B_i$). 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 $\sigma$-LFSR corresponds to a set of columns of the feedback matrices which are stored in a continuous block of memory. The state vector is now sampled one integer at a time and the columns of the $B_i$s 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 $\sigma$-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. CONCLUSIONS

In this paper, we have described a method of using $\sigma$-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-dependent random numbers to generate a random feedback configuration for the $\sigma$-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 keystreams generated by the proposed method are comparable to SNOW 2.0 from a randomness point of view.

ACKNOWLEDGEMENT

The authors are grateful to Prof. Harish K. Pillai, Department of Electrical Engineering, Indian Institute of Technology Guwahati for the valuable guidance and Associate Prof. Gaurav Trivedi, Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati for helping us 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 *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*. Bandung, Indonesia: IEEE, September 2016.

[2] M. Pistono, R. Bellafqira, and G. Coatriereux, “Secure processing of stream cipher encrypted data issued from iot: Application to a connected knee prosthesis,” in *41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Berlin, Germany: IEEE, July 2019.

[3] S. A. Jassim and A. K. Farhan, “A survey on stream ciphers for constrained environments,” in *21st Babylon International Conference on Information Technology and Science (BICITS)*. IEEE, 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, 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, “The t-class of sober stream ciphers,” *Unpublished manuscript. http://www. home.aone.net.au/qualcomm*, 1999.

[7] P. Ekdahl and T. Johansson, “Snow—a new stream cipher,” in *Proceedings of First Open NESSIE Workshop, KU-Leuven*. Citeseer, 2000, pp. 167–168.

[8] P. Ekdahl and T. Johansson, “A new version of the stream cipher snow,” in *International Workshop on Selected Areas in Cryptography*. St. John’s, Newfoundland, Canada: Springer, August 2002, pp. 47–61.

[9] G. G. Rose and P. Hawkes, “Turing: A fast stream cipher,” in *International Workshop on Fast Software Encryption*. Lund, Sweden: Springer, February 2003, pp. 290–306.

[10] P. Kitsos, G. Selimis, and O. Koufopavlou, “High performanceasic implementation of the snow 3g stream cipher,” *IFIP/IEEE VLSI-SOC*, pp. 13–15, 2008.

[11] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier et al., “Sosemanuk, a fast software-oriented stream cipher,” in *New stream cipher designs*. Springer, 2008, pp. 98–118.

[12] F. Xiu-tao, “Zuc algorithm: 3gpp lte international encryption standard,” *Information Security and Communications Privacy*, vol. 12, 2011.

[13] P. Ekdahl, T. Johansson, A. Maximov, and J. Yang, “A new snow stream cipher called snow-x,” *IACR Transactions on Symmetric Cryptology*, pp. 1–42, 2019.

[14] G. Zeng, W. Han, and K. He, “High efficiency feedback shift register: sigma-lfsr.” *IACR Cryptology ePrint Archive*, vol. 2007, p. 114, 2007.

[15] S. Krishnaswamy and H. K. Pillai, “On the number of linear feedback shift registers with a special structure,” *IEEE transactions on information theory*, vol. 58, no. 3, pp. 1783–1790, 2011.

[16] S. Krishnaswamy and H. K. Pillai, “On multisequences and their extensions,” arXiv preprint arXiv:1208.4501, 2012.

[17] O. Billet and H. Gilbert, “Resistance of snow 2.0 against algebraic attacks,” in *Cryptographers’ Track at the RSA Conference*. San Francisco, CA, USA: Springer, February 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 *International Workshop on Selected Areas in Cryptography*. Springer, 2003, pp. 222–233.

[19] K. Nyberg and J. Wallén, “Improved linear distinguishers for snow 2.0,” in *International Workshop on Fast Software Encryption*. Graz, Austria: Springer, March
2006, pp. 144–162.
[20] J.-K. Lee, D. H. Lee, and S. Park, “Cryptanalysis of seomunak and snow 2.0 using linear masks,” in International Conference on the Theory and Application of Cryptology and Information Security. 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.”
[22] S. Kiyomoto, T. Tanaka, and K. Sakurai, “K2: a stream cipher algorithm using dynamic feedback control,” in Secrypt, 2007, pp. 204–213.
[23] S. Ma and J. Guan, “Improved key recovery attacks on simplified version of k2 stream cipher,” The Computer Journal, vol. 64, no. 8, pp. 1253–1263, 2021.
[24] J. D. Golić, “Cryptanalysis of alleged a5 stream cipher,” in International Conference on the Theory and Applications of Cryptographic Techniques. 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,” Electronics Letters, 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 International Conference on Information Security and Cryptology, Seoul, Korea (Republic of), December 2004.
[27] S. Kiyomoto, T. Tanaka, and K. Sakurai, “K2: A stream cipher algorithm using dynamic feedback control,” in International Conference on Security and Cryptography (SECRYPT), Barcelona, Spain, July 2017.
[28] P. Ekdahl and T. Johansson, “Snow – a new stream cipher,” in 1st NESSIE Workshop, Heverlee, Belgium, November 2000.
[29] S. Krishnaswamy and H. K. Pillai, “On the number of special feedback configurations in linear modular systems,” Systems & Control Letters, vol. 66, pp. 28–34, 2014.
[30] H. Ahmadi and T. Eghlidos, “Heuristic guess-and-determine attacks on stream ciphers,” IET Information Security, vol. 3, no. 2, pp. 66–73, 2009.
[31] M. Hell, T. Johansson, and L. Brynielsson, “An overview of distinguishing attacks on stream ciphers,” Cryptography and Communications, vol. 1, no. 1, pp. 71–94, 2009.
[32] B. Debraize and I. Marquez Corbella, “Fault analysis of the stream cipher snow 3g,” in Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). Lausanne, Switzerland: IEEE, September 2009, pp. 105–112.
[33] M. S. N. Nia and A. Payandeh, “The new heuristic guess and determine attack on snow 2.0 stream cipher,” IACR Cryptology ePrint Archive, vol. 2014, p. 619, 2014.
[34] N. T. Courtois and B. Debraize, “Algebraic description and simultaneous linear approximations of addition in snow 2.0.” in International Conference on Information and Communications Security. Springer, 2008, pp. 328–344.
[35] M. Matsui, “Linear cryptanalysis method for des cipher,” in Workshop on the Theory and Application of Cryptographic Techniques. Springer, 1993, pp. 386–397.
[36] W. Meier and O. Staffelbach, “Fast correlation attacks on certain stream ciphers,” Journal of Cryptology, vol. 1, no. 3, pp. 159–176, 1989.
[37] V. V. Chepyzhov, T. Johansson, and B. Smeets, “A simple algorithm for fast correlation attacks on stream ciphers,” in International Workshop on Fast Software Encryption. Springer, 2000, pp. 181–195.
[38] F. Jönsson and T. Johansson, “Correlation attacks on stream ciphers over gf (2^n),” in IEEE International Symposium on Information Theory (ISIT), 2001, pp. 140–140.
[39] D. Wagner, “A generalized birthday problem,” in Annual International Cryptology Conference. Springer, 2002, pp. 288–304.
[40] Y. Todo, T. Isobe, W. Meier, K. Aoki, and B. Zhang, “Fast correlation attack revisited,” in Annual International Cryptology Conference. Springer, 2018, pp. 129–159.
[41] X. Gong and B. Zhang, “Fast computation of linear approximation over certain composition functions and applications to snow 2.0 and snow 3g,” Designs, Codes and Cryptography, pp. 1–25, 2020.
[42] G. Leander, E. Zeren, and P. Hawkes, “Cache timing analysis of lfsr-based stream ciphers,” in IMACryptology and Coding. Springer, 2009, pp. 433–445.
[43] B. B. Brumley, R. M. Hakala, K. Nyberg, and S. Sovio, “Consecutive s-box lookups: A timing attack on snow 3g,” in International Conference on Information and communications security. Springer, 2010, pp. 171–185.
[44] E. Yilmaz, “Two versions of the stream cipher snow,” Master’s thesis, Middle East Technical University, 2004.

A. SUBRATA NANDI has received his M. Tech. degree in computer science engineering from KIIT University, Bhubaneswar, India. He has worked as a Project Fellow in Mathematics department of National Institute of Science Education and Research(NISER), Bhubaneswar, India in a DST funded project. Currently he is working as a Ph.D. research scholar in Cryptology at CSE department of Indian Institute of Technology (IIT) Guwahati under the supervision of Dr. Srinivasa Krishnaswamy and Dr. Pinkai Mitra. His research area includes $\sigma$–LFSR, Word based LFSR and Cryptographic Boolean Function.
B. DR. SRINIVASAN KRISHNASWAMY has received his Ph.D. in Electrical Engineering from Indian Institute of Technology Bombay, Mumbai. He is currently an assistant professor at the Indian University of Technology (IIT) Guwahati, India. His research area includes cryptography and control systems.

C. BEHROUZ ZOLFAGHARI has received his Ph.D. in computer engineering from AmirKabir University of Technology, Tehran, Iran. He has worked as a postdoctoral research fellow at Indian University of Technology (IIT) Guwahati, India. His research area includes information theory, information-theoretic cryptography, hardware oriented cryptography and AI-assisted cryptography.

D. PINAKI MITRA is currently an associate professor at Department of Computer Science and Engineering, IIT Guwahati. He obtained his B. Tech in Computer Science and Engineering from Jadavpur University, Kolkata in 1987, India and his M. Tech in Computer Science and Engineering from Indian Institute of Science Bangalore, India in 1989. Finally, he obtained his Ph. D from Simon Fraser University, Canada in 1994. He worked on a project at Jadavpur University, department of Computer Science and Engineering. Subsequently he joined National Institute of Management, Kolkata, and served as an assistant professor. He joined IIT Guwahati in December, 2004. His research interest includes Cryptography, Network Security, Computer Graphics and Multimedia.