International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 14, No.1, 233-243 (Jul-23)

http://dx.doi.org/10.12785/ijcds/140120

Design and Analysis of a new Stream Cipher based on Ring FCSR Automaton

Praveen Kumar Gundaram[1,2], Appala Naidu Tentu[2], Neelima Guntupalli[1] and Nagendar Yerukala[2]

[1]*Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India - 522510*
[2]*C R Rao AIMSCS, UoH Campus, Gachibowli, Hyderabad, Telangana, India - 500046*

Received 16 Sep. 2022, Revised 30 Apr. 2023, Accepted 1 May. 2023, Published 1 Jul. 2023

Abstract: Feedback with Carry Shift Registers (FCSRs) are being most commonly used in the design of new stream ciphers. This is because the FCSRs have resistance to correlation and algebraic attacks because of their quadratic feedback [1], [2]. Two representations of FCSRs, namely, Galois and Fibonacci are quite popular. However, stream cipher designed using any of these representations is vulnerable to many attacks as proposed in [3], [4]. A new representation called the ring representation, which generalizes and avoids the weaknesses of both the representations of FCSRs has been proposed in [5]. Similar to other representations, the ring representations produce sequences that have high periods and entropy. The advantage of ring representation is that it results in faster diffusion and reduced implementation costs when compared to Galois and Fibonacci. This paper presents two designs of stream cipher based on ring representations. In both algorithms, the output is produced by linear filters that pass randomness tests. The ciphers proposed have undergone a thorough security analysis.

Keywords: Stream cipher, Feedback Shift Registers, Ring FCSR Automaton, *l*-sequence, Cryptanalysis, Transition matrix.

1. INTRODUCTION

Large-period LFSR sequences provide strong statistical properties. The disadvantage of its linear structure for cryptographic applications is that it may allow algebraic attacks. As a result, an LFSR-based cryptosystem should have a few difficult nonlinear blocks. As an alternative to LFSRs in stream cipher design, feedback with carry shift registers (FCSRs) has been suggested [6], [7], [8]. Feedback Shift Registers (FSRs) are recent primitives with wide applications in cryptography, coding theory, and information theory. Feedback Shift Registers are suitable for high-performance hardware and software applications. As shown in Figure- 1 and Figure- 2, FCSRs are represented in either Fibonacci or Galois modes [9]. In the Galois mode [5], one feedback bit impacts every cell, but in the Fibonacci mode [5], every feedback bit affects just one cell.

As shown in the Figure- 1 and Figure- 2 operations of the FCSR described more detailed in paper of Klapper and Goresky [1], set an odd positive integer q and let $q+1 = q_1 2^1 + q_2 2^2 + \cdots + q_r 2^r$ be the binary expansion of $q+1$, where $r = \lfloor log_2(q+1) \rfloor$ and $q_i \in \{0, 1\}$. The bits in the $q+1$ equation $\{q_1, q_2, \ldots, q_r\}$ of the 2-adic FCSR with connection integer q have r steps and feedback connection coefficients. Let $\{a_0, a_1, \ldots, a_{r-1}\} \in GF(2)$ denote the contents of the main shift register cells, and initial memory represents $m \in \mathbb{Z}$.

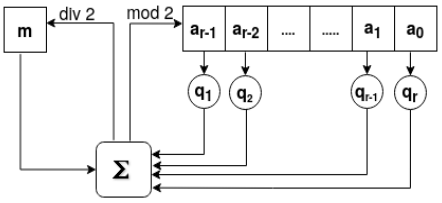

Figure 1. Fibonacci FCSR

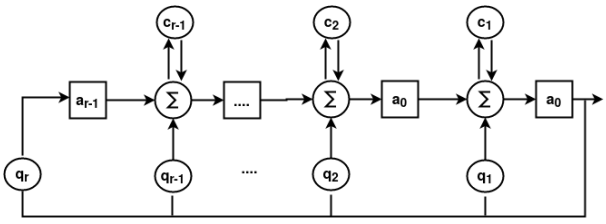

Figure 2. Galois FCSR

The main shift register and initial memory combination are known as FCSR. This mode is suitable for cryptosystems described in [4] since the majority of Fibonacci FCSR cells contain a linear transition function, reference [4]. Because of the dependence between the carriers and the

E-mail address: praveen.gundaram@gmail.com, naidunit@gmail.com, neelima.guntupalli80@gmail.com, nagendarmsc111@gmail.com

single feedback bit [10], a Galois FCSR might be modeled effectively during some clocks. This weakness, which has a non-zero likelihood of occurring, causes FCSRs to become LFSRized and makes all FCSR-based stream ciphers vulnerable to powerful attacks [11]. The keystream produced by the new proposed design has the period $|q|-1$ and also passes each test listed in the NIST test suite. So that the generated keystream is considered as a random sequence. It is resistant to several existing attacks.

Arnault et al,. [5] have invented a new FCSR representation termed the ring representation (which is a generalization of both Fibonacci and Galois representation) as shown in Figure - 3. This approach allows any cell to act as a feedback bit for another cell [12]. Thus, when this new ring mode is used, the attacks presented in [3], [11]. M. Hell et al [3] attack have no longer applicable because describing an attack requires Mbytes of received sequence with low complexity.

In this paper, we develop generalization methods using particular automata known as 2-adic automata. Initially build automata with inputs and outputs using matrix representations [5], [13]. Secondly, instead of using matrices with coefficients of $\{0,1\}$ as was the case with a standard FCSR, We use matrices that contain coefficients that are 2-adic integers. The practical implications of such automata, including their size and memory needs, are then covered. Finally, we discuss the possible uses of these automata in the development design of hardware- and software-based stream ciphers. Stream ciphers could continue to play an important role in constrained IoT application where high throughput remains critical and resources are very restricted [14].

The structure of this paper is as described in the following: Section 2 describes the characteristics of 2-adic integers and generalized the ring-FCSR automata using matrix representations and filter functions. In Section 3, we propose a new automaton design. In Section 4, we provide the security analysis and requirements of Ring-FCSRs. In Section 5, we proposed a word-based automaton design with an algorithm. Using such FCSRs in cryptography applications is described in this article. Conclusion remarks have been discussed in section 6.

2. Preliminaries

A. 2-adic Integer and Period

The power series $s=\sum_{i=0}^{\infty} s_i.2^i$, is used to formally represent a 2-adic integer. Let $\mathbb{Z}_2$ [5] denoted the set of 2-adic integers. In contrast to addition and multiplication in $\mathbb{F}_2$, they are carried out in $\mathbb{Z}_2$ by adding the carries to the higher order terms. Positive integers are represented as $00000\ldots s_{n-1}\ldots,s_0$, where $s_0,s_1,\ldots,s_{n-1}$ corresponds to the binary expansion of the integer. Negative integers are represented using 1's complement. Only odd integers have inverses in $\mathbb{Z}_2$. From this, it can be concluded that $\mathbb{Z}_2$ contains a rational number of the from p/q, with q being odd.

There exists a bijection between the set of eventually periodic binary sequences and a set of 2-adic integers of the form p/q where $p,q\in\mathbb{Z}$ and q is odd. There also exists a bijection between the set of periodic binary sequences and set of 2-adic integers of the form p/q where $|p|\le|q|$ and $pq\le 0$.

Let p/q be the 2-adic integer corresponding to the binary sequence $S=(s_n)_{n\in\mathbb{N}}$. The order of an element 2 in $\mathbb{Z}^*_{|q|}$ is the period of this sequence S. Further, the sequence S is said to be l-sequence, if 2 is the generator of $\mathbb{Z}^*_{|q|}$.

B. Ring FCSR

A ring FCSR [5] consists of two components

1) A main shift register (m) of length n. Each cell $(m_i,\ 0\le i\le n-1)$ contains {0 or 1}.
2) A carry register (c) also of length n. Each cell $c_i\in\mathbb{Z}$ where $0\le i\le n-1$.

The ring FCSR is updated as follows [15]:

$$\left.\begin{aligned} m(t+1) &= T.m(t)+c(t) \mod 2\\ c(t+1) &= T.m(t)+c(t) \operatorname{div} 2 \end{aligned}\right\} \quad (1)$$

$T=(a_{i,j})_{1\le i,j<n}$ where $a_{i,j}$ is the element at the i^{th} row and j^{th} column of $n\times n$ matrix with coefficients 0 or 1 in $\mathbb{Z}_2$ called transition matrix, which is of the following form:

$$T=(a_{i,\ j})_{0\le i,j<n} \text{ with } a_{i,j}=\begin{cases}1 & \text{if } m_j \text{ is used to update } m_i,\\ 0 & \text{otherwise}\end{cases}$$

$$T=\begin{pmatrix} * & 1 & & & & \\ & * & 1 & & (*) & \\ & & * & 1 & & \\ & & & \ddots & \ddots & \\ & & (*) & & * & 1\\ 1 & & & & & * \end{pmatrix}$$

Since the main register of a ring FCSR is a shift register, $a_{i,\ i+1 \bmod n}=1\ \forall\ 0\le i\le n$ by definition to preserve the ring structure of the automaton.
Note: X div $2=(\ X-(X \bmod 2)\ )/2$

Theorem 1: The series $M_i(t)$ observed in the cells of the main register are 2-adic [5] expansion of p_i/q with $p_i\in Z$ and with $q=det(I-2T)$.

Lemma 2: q is referred to as the connection integer [5] of the FCSR. If $q=det(I-2T)$ is prime and if the order of 2 in $\mathbb{Z}/q\mathbb{Z}$ is maximal, then each M_i is an l-sequence.

A typical example of Ring FCSR [5] is described in Figure- 3. In this example, the symbol $\boxplus$ refers to addition

with carry. Which is typically implemented using Full Adder [16]. If suppose a,b and c are the input to the full adder and sum and carry are the outputs. then the equations sum = $a \oplus b \oplus c$ and carry = $a.b \oplus b.c \oplus c.a$. At the next transition, the carry part of the output is delayed and given back as input. Prime number $q = -347$ has the maximal order of 2 in $\mathbb{Z}/q\mathbb{Z}$ and is a $|q|$ = det(I-2A). As a result, the FCSR produced an l-sequence in Figure- 3 as its output sequence.

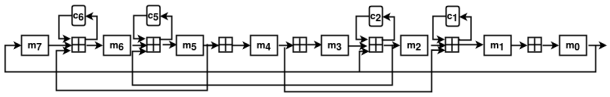

Figure 3. Ring FCSR with connection integer (q=-347)

C. Design Criteria: Transition Matrix T

The transition matrix T is any random $n \times n$ matrix that must meet the following criteria [8]:

i The matrix must have a general weight of $n + l$ and be made up of the entries $\{1, 0\}$. Let $n+l$ be as small as possible. To secure a strong nonlinear structure, it generally requires between $n/2 - 5$ and $n/2 + 5$ willing feedbacks (l).

ii The number of feedbacks (1's) in each row/column is less than or equals two. By boosting the number of cells that the feedback may reach, this condition enables improved diffusion. In addition, it offers uncorrelated carries and a fan-out with a diameter of 2.

iii Matrix should ensures a non-degenerated matrix(T). $\log_2(q) \geq n$ where $q = \det(I - 2T)$ and $\det(T) \neq 0$; So that sequence output by FCSR is an l-sequence.

iv Connection integer q is prime; $2^{|q|-1} \cong 1 \mod q$

v The diameter of graph G constructed by the transition matrix T as its adjacency matrix should be minimum for lower diffusion delay.

Algorithm- 1 generates a $n \times n$ transition matrix T that satisfies all of the design requirements described in this section.

Algorithm 1 : Construction of Transition Matrix

Input: Pre-constructed Transition Matrix $\mathbb{T} = (a_{i,j})_{0 \leq i,j < n}$
Output: Full constructed Transition Matrix($\mathbb{T}$) with connection integer(q) is prime.
Initialisation :
Choose $\boldsymbol{n}$ as size of the matrix size.
$count \leftarrow 0$
for $t_1 = 0$ to n^2 **do**
  Get preconstructed Transition Matrix $T = a_{i,i+1 \ mod \ n} = 1 \ \forall \ 0 \ \leq \ i < n$.
  Choose an optimal $l \leftarrow [n/2 - 5, \ldots, n/2 + 5]$
  Randomly select the number of feedbacks l in the Transition Matrix.
  Choose feedback(l) position at most one in each row & column. (as a row-pos, col-pos)
  for $t_3 = 0$ to l **do**
    T[row-pos, col-pos] $\leftarrow$ 1
  end for
  $q \leftarrow \det(I - 2T)$
  if $(det(T) \neq 0)$ **then**
    if det(q) is prime **and** $2^{|q|-1} \cong 1 \mod q$ **then**
      if $\log_2(q) \geq n$ **then**
        Consider the Transition Matrix T and q value.
        $count \leftarrow count + 1$
      else
        Does not satisfy $log(q)$ value at this stage.
      end if
      q value may not be prime and $2^{|q|-1} \not\cong 1 \mod q$.
    end if
    $det(T) = 0$
  end if
end for
for $T = 1$ to $count$ **do**
  Consider a minimum number of feedbacks.
  Filter out the Transition Matrix whose graph has a minimum diameter(d).
end for
return T

D. Design Criteria: Linear Filter

In the case of stream cipher using LFSRs, the output is produced after combining the LFSR state using a highly non-linear boolean function. For FCSRs, since the feedback function is quadratic the output can be produced after filtering using a linear Boolean function [17].

For a fixed-size input, linear Boolean functions provide the maximum protection against correlation attacks. Both software and hardware design development variants are the most productive Boolean operations in terms of timing and circuit complexity. Therefore, a filter F to produce an output of one bit is a linear $GF(2)$ operation of the form

$$F : GF(2)^n \rightarrow GF(2), \quad F(x_1, \ldots, x_n) = \oplus_{i=1}^{n} f_i x_i, f_i \in GF(2)$$

At a particular time instant, where $x_1, \ldots, x_n$ are the elements of the main shift register of the size n. A filter F can also be viewed as a vector $(f_1, \ldots, f_n)$ of size n over $GF(2)$. It can also be viewed as an integer $F = \sum_{i=1}^{n} f_i 2^{i-1}$.

At each clock Ring Filter, FCSR produces an output of u number of bits, in the general output, should be between 1 to $n/16$ to provide a challenging invertibility of the filter. Filters are produced output should ensure good statistical properties and a long period.

E. Design Criteria: Key/IV setup

A sufficient level of confusion and diffusion of the initial state bits is often achieved by the initialization process (Key/IV configuration). A common method of initialization is to repeatedly clock the Keystream Generator without creating keystream bits [18].

The Key/IV configuration process must adhere to the following requirements:

i In order to prevent the time-memory-data-trade-off (TMTO) attack, the entropy provided by the key should be preserved (in other words the state should not lose entropy)

ii To resist a straightforward key recovery attack, the transformation given by the Key and IV configuration process must be complex to invert.

iii In order to avoid re-synchronization attacks, the number of clocks the generator runs before producing any output should ensure better diffusion.

3. Proposed Stream Cipher Design

The new stream cipher (Ring-Filter-FCSR) mainly consists of two components, i.e, Ring FCSR and linear filter. Ring FCSR is used for maintaining the state of the stream cipher. A linear filter is used for producing the output of the stream cipher as shown in the Figure- 4. At each iteration, the state gets updated with the help of the transition matrix.

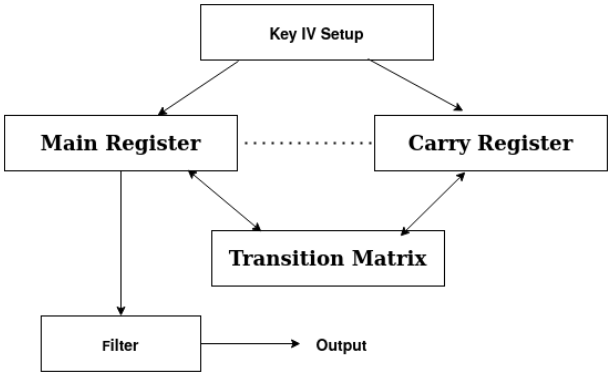

Figure 4. Block Diagram of the Stream Cipher.

A. Design of Ring-Filter-FCSR-8-bit (RFF8)

The proposed Ring-Filter-FCSR-8-bits (RFF8) stream cipher as the main register and transition matrix has to satisfy theorem- 1 and lemma- 2. In RFF8 stream cipher main register contains the size of $n = 128$. RFF8 structure takes a Key(K), and Initial Vector(IV) of the sizes $96, 32$ respectively, and also produces an output (u) of an 8-bit word. In this case, d is the diameter of the transition graph.

The design consists of two steps: 1. Key/IV setup and 2. Keystream generation stage. The parameters for the proposed stream cipher design are chosen based on the design criteria.

- The main register size $n = 128$.
- Size of the transition matrix (T) is 128×128.
- The $Key(K) = 96$ and $IV = 32$.
- The number of feedbacks (l) is 65.
- Output(u) $= 8 - bits$.
- Diameter (d) $= 24$.
- $q = 53141674284678874070058934030498056420 1$

The transition matrix $T = (a_{i,j})$, $0 \le i, j \le 127$ is $n \times n$ over the GF(2), is of the following form:

$$\text{Transition matrix (T)} = \begin{cases} a_{i,i+1(\text{mod } 128)} = 1 & \text{for } 1 \le i \le 127, \\ a_{i,j} = 1 & \text{for (i, j)} \in \mathbf{U} \end{cases}$$

$$U = \begin{pmatrix} (0,44) & (3,49) & (4,45) & (5,37) & (6,86) & (10,70) & (11,14) & (15,17) \\ (19,98) & (21,71) & (24,93) & (25,82) & (30,41) & (31,125) & (33,22) & (34,18) \\ (35,66) & (36,26) & (37,35) & (38,42) & (39,32) & (40,33) & (43,106) & (44,62) \\ (45,100) & (46,115) & (49,99) & (50,2) & (51,59) & (52,57) & (53,113) & (54,12) \\ (58,69) & (62,9) & (64,15) & (65,121) & (66,102) & (70,112) & (71,60) & (76,79) \\ (79,126) & (83,16) & (86,89) & (89,55) & (92,97) & (93,61) & (98,78) & (99,52) \\ (100,95) & (101,119) & (102,11) & (106,94) & (107,85) & (109,120) & (110,80) & (112,68) \\ (113,0) & (114,101) & (115,67) & (116,51) & (117,38) & (121,114) & (122,64) & (124,31) \\ (125,24) & & & & & & & \end{pmatrix}$$

$|U|$ is the number of willing feedbacks l. The transition matrix T is represented using the graph given in Figure 5. An edge (i, j) indicates that register m_j is used to update register m_i

Key / IV setup:

The main shift register of 128 bits denoted by $\{m_{127}, m_{126}, \ldots, m_1, m_0\}$ and Key(K) of length 96 bits denoted by $K = k_{95}||k_{94}||\ldots||k_1||k_0$ and Initial Vector(IV) of length 32 bits denoted by $IV = v_{31}||v_{30}||\ldots||v_1||v_0$. Key and Initial Vector(IV) are loaded into the main shift register using the below-mentioned *mix-function*. Initially carry(c) register set with 0 as shown in the below equation.

Loaded m(0) amd c(0) from the *mix-function*(Key, IV) is given below.

$$\left.\begin{aligned} m(0) &= k_{95}||k_{94}||\ldots||k_1||k_0||v_{31}||v_{30}||\ldots||v_1||v_0 \\ c(0) &= 0 \end{aligned}\right\} \quad (2)$$

Update the main($m(t)$) and carry($c(t)$) registers are (by equation-1) using clock function and produce pseudo-random word of length 8, S_i ($0 \le i < 16$) in each iteration with the help of filter function. The main register $m(t)$ reset with the producing $S_{15}||\ldots||S_1||S_0$ values and $c(t) = 1$. The Ring-FCSR is clocked $d + 4$ times without producing any output. where d is diameter of the transition graph such that d = max{distance(m_i, m_j), $0 \le i, j \le 127$} where each m_i, $\forall$

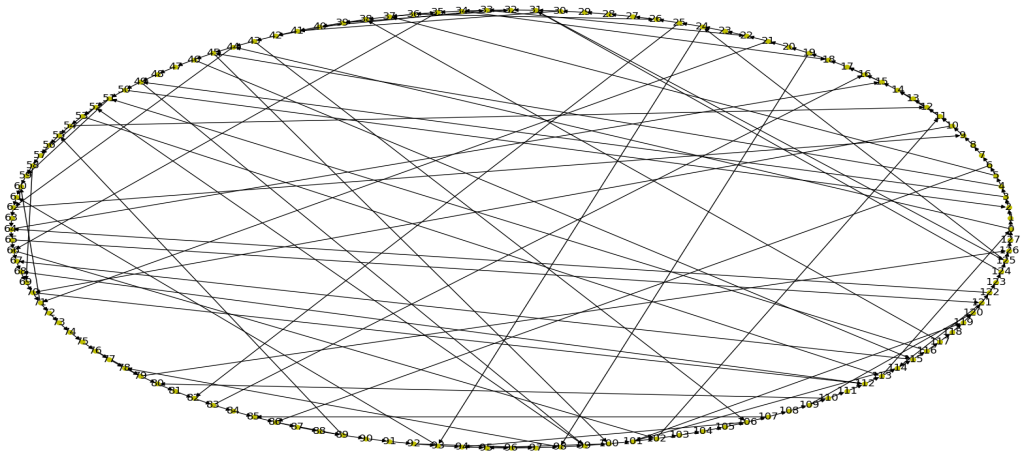

Figure 5. State update diagram/graph of RFF8

i is a cell in the main register.

i The main register $m(t)$ is reinitialized as follows: $m(t) = (S_{15}\| \ldots \|S_1\|S_0)$

ii The carry register is reinitialized to zero: $c(t) = 1$.

iii Update the registers $d + 4$ times without producing any output.

Key Stream generation:

After the initialization stage, the ring FCSR is clocked regularly as shown in the Figure- 5 and it produces 8-bits keystream. To produce a ciphertext we need to XOR keystream bits with the plaintext message.
Filters: In the RRF8, design the filters $F_0, \ldots, F_7$ that are used given below.
$F_0 = \{0, 19, 35, 45, 58, 79, 100, 113, 125\}$
$F_1 = \{3, 21, 36, 46, 62, 83, 101, 114\}$
$F_2 = \{4, 24, 37, 49, 64, 86, 102, 115\}$
$F_3 = \{5, 25, 38, 50, 65, 89, 106, 116\}$
$F_4 = \{6, 30, 39, 51, 66, 92, 107, 117\}$
$F_5 = \{10, 31, 40, 52, 70, 93, 109, 121\}$
$F_6 = \{11, 33, 43, 53, 71, 98, 110, 122\}$
$F_7 = \{15, 34, 44, 54, 76, 99, 112, 124\}$
Each filter F_i, $(0 \leq i \leq 7)$ outputs the XOR of bits taken from the specific main register cells of ring FCSR. i^{th} $(0 \leq i \leq 7)$ keystream bit is the result of applying the filter F_i on the state of the ring FCSR.

B. Design Performance Analysis

For the purpose of evaluating the throughput of our RFF8 design, we have chosen other stream ciphers and compared the performance of the main parameters. Calculate the execution time to produce keystream in the time of bits transmitted per second. Implementations are carried out using C, an Ubuntu 18.04 64-bit operating system with 8GB RAM, Intel Core i7 CPU 860 @ 3.2 GHz × 12 processors. We compared with the F-FCSR-32 benchmark suite and obtained the keystream speed results of 11.92 bits per second but our proposed RFF8 design takes 0.032 MBps to produce keystream results.

4. Security analysis of proposed design

The new designs are immune to existing attacks used to evaluate their security. Various cryptanalytic techniques have been applied to the proposed stream cipher. It is resistant to several attacks such as distinguisher attacks [19], and correlation attacks, and the keystream generated by it has good statistical properties like large period, high entropy, and high linear complexity [20]. It has been shown that retrieving the key or initial vector given the keystream is extremely difficult based on the correlation tests.

A. Distinguisher Attack

The fact of linear combinations between internal automaton states that occur with a biased probability can be used as a basis for distinguishing attacks [19]. The existence of such relations seems implausible given the presence of carry cells. Since we were unable to locate any, we think that none of them exists.

B. Correlation Tests on proposed stream cipher

Correlation test between Keystream and Key: This test calculates the correlation between key and generated keystream [21]. The adversary can use it to derive the key or reduce the search space of an exhaustive search attack.

The experiment in Table-I uses $2^{20} - 1$ random keys and zero initial vector as input and generates the n-bit key stream of RFF8. Calculate the Hamming Weight of the keystream xor key (i.e. $W_i = HW(Z_i \oplus K_i)$). The RFF8 is secure if the hamming weights probability distribution is binomial. Use the binomial distribution to calculate the probabilities of these weights. Hamming weights have been grouped so that each group roughly has the same probability. Perform the $\tilde{\chi}^2$-Test [22]. The outcome is p_value, by this value, we can say that the design is secure the p_value should be ≥ 0.01.

TABLE I. Correlation test between Keystream and Key

KS & Key test	OBSERVED	EXPECTED
Group(0-59)	223109	223576.8
Group(60-62)	190919	191138.4
Group(63-65)	219465	219144.3
Group(66-68)	191223	191138.4
Group(69-128)	223859	223576.8
Total	1048575	1048576
	p-value	0.71
	$\tilde{\chi}^2$	2.09

Correlation test between Keystream and IV: This test calculates the correlation between the initial vector and generated keystream [21]. The existence of a correlation implies that the part of the keystream can be generated without the knowledge of the key. The IV setup must be redesigned if a stream cipher fails this test.

The experiment in Table-II uses a random key and $2^{20} - 1$ random initial vector as input and generates the n-bit key stream of RFF8. The following Table- II gives the correlation [21] between keystream and *IV* and keystream in results (Groups are formed using hamming weights as like a Table- I).

TABLE II. Correlation test between Keystream and *IV*

KS & IV test	OBSERVED	EXPECTED
Group(0-59)	222810	223576.9
Group(60-62)	191313	191138.4
Group(63-65)	219283	219144.3
Group(66-68)	191168	191138.4
Group(69-128)	224001	223576.8
Total	1048575	1048574.8
	p-value	0.45
	$\tilde{\chi}^2$	3.68

C. Randomness

Randomness tests examine very closely whether the provided binary sequence represents a truly random sequence [23]. Typically, a test statistic is used to do this. A true random sequence of data can be tested by using this test statistic, which pursues a specific probability distribution $\mathcal{P}$. The test-statistic value and the probability that it will take the value $\mathcal{V}$ under the probability distribution *Prob* are computed for the supplied binary sequence to be tested (This probability is known as the p-value). In terms of true randomness, the binary sequence can be considered if the p-value exceeds 95% or 99% level of significant value (where α is 0.01 or 0.05).

NIST [23] and DIEHARD test suites are used to perform randomness tests. NIST has devised fifteen randomness tests to evaluate the randomness stream. The sequence produced by a PRNG is the input for a separate statistical test in the NIST test suite, which produces a result showing whether the sequence is random or not. Consider the claim that PRNG-generated sequences are random according to statistical tests. We do statistical tests on the output sequence to identify if the stream ciphers act as pseudo-random bit generators. Keystream of size one lakh and ten lakhs has been produced from newly designed stream cipher. We applied NIST tests on these keystreams. The results have been tabulated in Table III.

TABLE III. NIST Test Suite result of 1-lakh & 10-lakhs keystream bits

TEST NAME	Result Suc/Fail	*p_value* (1Lakh bit-stream)	*p_value* (10Lakh bit-stream)
1. Runs	Success	p=0.687158	p=0.925766
2. Block Frequency	Success	p=0.487911	p=0.248277
3. Frequency	Success	p=0.379340	p=0.324133
4. Discrete Fourier Transform	Success	p=0.749568	p=0.818546
5. Rank	Success	p=0.687158	p=0.925766
6. Longest Run of Ones	Success	p=0.967290	p=0.191964
7. Universal Statistical	Success	–	p=0.40625
8. Overlapping Template Matchings	Success	p=0.577567	p=0.856626
9. Non-overlapping Template Matchings (Successes out of 148 Sub-Tests)	Success	146/148	147/148
10. Approximate Entropy	Success	0.210755	0.027123
11. Serial	Success	p_1=0.424892 p_2=0.077674	p_1=0.470194 p_2=0.276086
12. Linear Complexity	Success	0.547553 $\tilde{\chi}^2$=4.970881	p=0.296024 $\tilde{\chi}^2$=7.276588
13. Random Excursions Variant	N/A	Insufficient cycles	Insufficient cycles
14. Random Excursions	N/A	Insufficient cycles	Insufficient cycles
15. Cumulative Sums	Success	p_1=0.727154 p_2=0.277757	p_1=0.255008 p_2=0.478352

Additionally, we examined the sequences produced for selected keys with fixed patterns, shown in Table- IV.

D. Linear Complexity

The linear complexity of a sequence output by a filtered Galois FCSR has been analyzed in [24]. The same argument is applicable to ring FCSRs because of Theorem 1.

From Theorem 1, the sequence observed in each cell of

TABLE IV. Test Vector of RFF8 with various Key, *IV* settings

Input parameters (Key, IV)	Output (Keystream)
Key=0x00000000000000000000000000 IV=0x00000000	0x6c33464eabdb9602e565174e7ce99f3e
Key=0x80000000000000000000000000 IV=0x80000000	0x140bd118dbebe73da0da85fad1b3bf0d
Key=0xc27dfea157408b90eb2afe51 IV=0x4a9f7c26	0x93e5d97d4f0cbbbdedc13c8472ece42d
Key=0xffffffffffffffffffffffff IV=0xffffffff	0xaa24e158e3376e02f1380605ac474834

the main register is the 2-adic expansion of p_i/q. The linear complexity of such a sequence is high (close to that of a random sequence). the output keystream is the XOR of these sequences. Since each sequence is high linear complexity, the XOR is also of high linear complexity [20].

E. Cryptanalysis of Multiple Filters

Let $F_1, \ldots, F_u$ be u filters to be used on the exact state of the main register (Every filter selects a subset of bits from the main register's state and produces a one-bit output). The following conditions need to be ensured while designing multiple filters:

- Every filter must be immune against the 2-adic attack.
- Each of the u filters must be linearly independent (when viewed as vectors) so that any dependency on the u outputs is avoided.
- 2^u filters can be obtained by all possible linear combinations of the u filters. Each such filter must be immune against a 2-adic attack.

The multiple filters $F_1, \ldots, F_u$ will generate a binary linear code C.

- The dimension of the code C is u because the filters are linear independent.
- Suppose k_F be the smallest integer for $F \in C$, where F is treated as an integer, and k_F is the condition that $2^{k_F} > F$. The values of k_F must have a minimal over C that is as large as feasible. (so that the cost of a 2−adic attack is no better than an exhaustive attack).
- Because it is feasible to build a filter on d cells of the main register from a code word of weight d, the C code with a minimal distance d should be avoided. A code C satisfying $d \geq 6$ is recommended.

The basis of any u-dimensional code of length n whose minimum hamming distance is greater than 6 and whose $minK_F$ is as large as possible can be used as u filters.

F. Correlation Attack

Consider two binary sequences, $S_i = (s_i)$ and $S_j = (s_j)$. The integer $\alpha(n) = \sum_{i=0}^{n-1}(-1)^{s_i \oplus s_j}$ is the correlation between the n first bits of S_i and S_j. $\alpha(n)$ is a random variable of mean $m = 0$ and variance $\sigma^2 = n$ for two random sequences S_i and S_j.

As explained in the following paragraphs, the correlation attack is based on the combined LFSRs (proposed in [25]). F will act as the generator's combining Boolean function. Assume that one of the inputs has a value that is not independent of the output of F. This input is connected to an LFSR which the attacker selects as a start point, to determine the correlation between the sequence produced by the LFSR connected to this input and the series specified by the generator. $|\alpha(n)|$ is likely to be quite little if the LFSR's setup is done incorrectly. It's probable that $|\alpha(n)|$ will be higher if the initialization is appropriate it means that matches the key.

This method can be extended if input(t) and output(F)'s xor are dependent on each other. Since there will be no correlation between any set's combination of at max input/output, the boolean functions utilized in these generators have to be robust of order t as large as possible. Similar to this, it is feasible to identify certain linear relationships between the intermediate states of the automaton for a binary filter LFSR with like a Boolean function F. It is feasible to create an extension of the pre-correlation attack that used a choice of correlations (proposed in [25]).

The application of this approach on a ring FCSR faces two main challenges [24].

- Firstly, Automaton's filtering function, which has l input parameters, is linear. Thus a function is balanced and has no connection between its output data and any total of at most $l-1$ of its input data, making it $l-1$ adaptable. The approach in that method is more challenging than the Bruteforce attack.
- The second is that because the transition function is polynomial (degree more than two), the dependencies between the cells of an FCSR automaton are nonlinear. The discovery of linear dependencies is hard.

5. Design of Ring-Filter-FCSR-32-bit (RFF32)

We propose a word-based Ring FCSR [26], [27], [28]. The major aim of our approach is to develop an optimized software-based design using a transition matrix (A) [29]. Block matrix(A) is obtained from $M_{n/w}$. ($M_w(F_2)$) of dimension n, split into blocks with sizes $r = n/w$ across F_2, represented by $A_{i,j}$.

The major aim of our approach is to develop an optimized software-based design using a transition matrix (A). Block matrix (A) is obtained from the array $M_{n/w}$. ($M_w(F_2)$)

of dimension n, split into blocks with sizes $r = n/w$ across F_2, represented by $A_{i,j}$. FCSR is defined by block(r), which we name a Ring-Filter-FCSR-32bit (RFF32). A Ring automaton is made up of a main register denoted by $M = (M_0, M_1, \ldots, M_{r-1})$ where M_i is a w-bit word and a carry register denoted by $C = (C_0, C_1, \ldots, C_{r-1})$. The block matrix(A) is of the following:

$$A = (A_{i,j})_{1 \le i,j \le n} \quad \text{with}$$

$$A_{i,j} = \begin{cases} '*' & \text{if } A_j \text{ is used to update the } A_i, \\ O & \text{otherwise} \end{cases}$$

In this case, $'*'$ represents for apply the itr times left shifts/right shifts operation of $w \times w$ matrix or else zero matrix and O represents zero matrix of the same $w \times w$ size. The transition matrix is of the following form

$$A = \begin{pmatrix} * & I_w & & & & \\ & * & I_w & & (*) & \\ & & * & I_w & & \\ & & & \ddots & \ddots & \\ & & (*) & & * & I_w \\ I_w & & & & & * \end{pmatrix}_{r \times r}$$

where I_w represents a binary identity matrix with dimensions of $w \times w$, and each row and column has r blocks..

These Ring FCSRs are appropriate for software development procedures since the super diagonal is full of identity matrices and faster to implement using memory copy or, in some conditions like the sliding window approach.

Let X be a w-bit word where $(x_0, \ldots, x_{w-1})$ such that x_{w-1} is most significant bit, (x_0) is least significant bits. The following operations may be defined for w-bit word: a) copy the memory of matrix, apply left/right shifts on matrix. The following definition applies to the w-bit words matrix that implements the left/right shifts (which are indicated by SL/SR).

$$\mathrm{SL} * (x_0, \ldots, x_{w-1})^t = (x_0, \ldots, x_{w-1}, 0)^t$$

$$\mathrm{SR} * S(x_0, \ldots, x_{w-1})^t = (0, x_0, \ldots, x_{w-2})^t$$

The blocks $A_{i,j}$ used to build the matrix A in this instance is equivalent to SL^a, SR^b where a and b represent the number of shifts.

For the implementation of software, design size is a major part as compared with hardware automaton design [28]. We work with single-bit operation in hardware, but in software at a time, we operate multiple bits as a word. It is not usual to work with a single bit and usually takes more operation instructions. Word sizes can range from $8-bits$ to $64-bits$ or more, depending on the processor's architecture.

A. Choice of block Transition Matrix

In the proposed stream cipher number of blocks r=8 and w=16. The block matrix [27] has been chosen using the algorithm- 2.

Algorithm 2 : Construction of word-based Transition Matrix

Input: The word size is w, and the Word ring FCSR length is n to seek with $n//w$.

Output: Transition Matrix($\mathbb{A}$) with word(w), feedback(l) and block matrix (A_w) (left/right) shift and also connection integer is prime.

Initialisation :

Choose n as size of the matrix size.
count = 0
for $t_1 = 0$ to n^2 **do**
  A=$(A_{i,j})_{0 \le i,j < n/w}$

$$\text{where } A_{i,j} = \begin{cases} I_w & \text{if j} \equiv \text{(i+1) mod n/w} \\ O & \text{otherwise} \end{cases}$$

  Construct $A_{n \times n} \leftarrow I_w$ & O_w of block matrices.
  Choose feedback(l) $\leftarrow$ [n/2-5, ..., n/2+5]
  select shift(f) $\leftarrow \{1, \ldots, w\}$, and $d \approx n/16$
  for $t_2 = 0$ to l **do**
    row, col, f = random([0,n/w,f])
    SL^f or $SR^f = A_w = I_w[row, col]^f$
  end for
  for $t_3 = 0$ to n **do**
    Load all SL and SR into A (update A_w into A)
  end for
  $q \longleftarrow \det(I - 2A)$
  if $(det(A) \neq 0)$ **then**
    if det(q) is prime **and** $2^{|q|-1} \cong 1 \mod q$ **then**
      if $\log_2(q) \ge n$ **then**
        Table[$count$] = A
        $count \leftarrow count + 1$
      else
        Does not satisfy $log(q)$ value at this stage.
      end if
      q value may not be prime and $2^{|q|-1} \not\equiv 1 \mod q$.
    end if
    $det(A) = 0$
  end if
end for
for $t_4 = 1$ to $count$ **do**
  if optimal(l) and min(d) **then**
    A = Table[t_4]
    Considered an optimal number of feedbacks.
    Filter out the minimum diameter(d) of A.
  end if
end for
return A

The word-based ring FCSR representing the transition

matrix (A) is given below

$$A = \begin{pmatrix} O & I_{16} & SL^7 & O & O & O & O & O \\ O & O & I_{16} & O & O & SR^8 & O & O \\ O & O & O & I_{16} & O & O & O & SL^5 \\ O & O & O & O & I_{16} & O & SL^5 & O \\ SR^6 & O & O & O & O & I_{16} & O & O \\ O & O & O & SL^1 & O & O & I_{16} & O \\ I_{16} & O & O & O & O & O & O & O \end{pmatrix}_{8\times 8}$$

Where O is a zero matrix with an order of 16, and I_{16} is an identity matrix. Let I_{128} represent the Identity matrix of order 128. It follows that the $q = det(I_{128} - 2A)$, q = 266842189815334043341065566729791008921 7, is a prime and $2^{|q|-1} \cong 1 \mod q$. The number of feedbacks (l) is 64. diameter of the graph (d) = 20. The related Word Ring FCSR automaton generates a sequence with a period of 26684218981533404334106556672979100892 16 as a result.

B. Update function

Let the main register's content blocks of words to be denoted as $M(t) = (M_0(t), \ldots, M_7(t))$ and carry register be denoted as $C(t) = (C_0(t), \ldots, C_7(t))$ at the time t. The size of the transition matrix T is 128 that is 8*16bits = 128bits. Ring Automaton always produces a fixed-sequence since the operations SR^6, SL^7, SL^1, SR^8, SL^5 and SL^5 on m_0, m_2, m_3, m_5, m_6, and m_7, respectively. The main register state is being updated as shown in Figure 6.

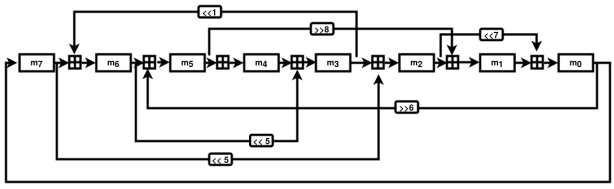

Figure 6. State update Word Ring FCSR (WRFF32)

Ring automaton RFF32 design contains Main register(M) and Carry register(C). M has r (= 8) number of blocks and, in each block stores w (= 16) bit word integer. Suppose main register initial denoted by $M = (M_0, M_1, \ldots, M_7)$ where M_i is a 16-bit word and a carry register denoted by $C = (C_0, C_1, \ldots, C_7)$. Figure 7 shows the state update process.

The word-based ring FCSR state will update as follows:

$X_0 = M_1 + (M_2 \ll 7) + C_0$
$X_1 = M_2 + (M_5 \gg 8) + C_1$
$X_2 = M_3 + (M_7 \ll 5) + C_2$
$X_3 = M_4 + (M_6 \ll 5) + C_3$
$X_4 = M_5 + C_4$
$X_5 = M_6 + (M_0 \gg 6) + C_5$
$X_6 = M_7 + (M_3 \ll 1) + C_6$

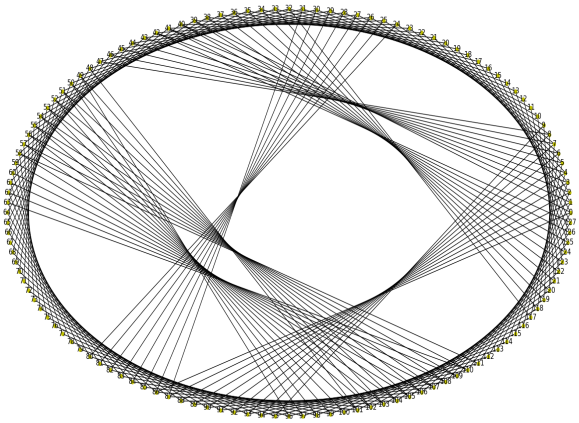

Figure 7. State update diagram/graph of RFF32

$X_7 = M_0 + C_7$

$$\left.\begin{aligned} M = X_i \ mod\ 2^{16} \quad \forall \quad 0 \le i < 8 \\ C = X_i \ div\ 2^{16} \quad \forall \quad 0 \le i < 8 \end{aligned}\right\} \tag{3}$$

C. Extraction function

The extraction function is made up of an extract Z-function that takes 128 bits of input, which has 8 register cells, each cell holds 16-bits memory, and produces an output of 32 bits from each iteration. The entire extraction function operates more formally with m_0 to m_7. The 32-bit word is computed and stored in $Z(t)$ at time t as follows:

$$Z(t) = (M_0||M_1) \gg 2 \ \oplus \ (M_2||M_3) \ll 1 \ \oplus \ (M_4||M_5) \lll 12 \ \oplus \ (M_6||M_7) \lll 25 \ \Big\} \tag{4}$$

D. RFF32 Design: Key Stream Generation:

RFF32 design categories into two stages Key/IV setup and keystream generation. The parameters of the Key length (key) = 96, Initial Vector (IV) length = 32 in RFF32 stream cipher. Output bits in each iteration (u) = 32. Initially load Key, and Initial Vector into the main register and carry register set with zero. During warmup, we have to update the state 4 times using equation 3 mean while extracting the 32-bit using equation 4. In the next stage, M has to load with extracted 128-bit output and carry set one (C = 1) and update the register 5 times, discard the produced output bits. Each iteration updates the state to generate the keystream bits of 32-bit. This design is undesirable for some cryptographic applications.

We examined the test vector RFF32 design sequences produced for selected keys with fixed patterns, shown in Table- V.

TABLE V. Test Vector of RFF32 with various Key, *IV* settings

Input parameters (Key, IV)	Output (Keystream)
Key=0x00000000000000000000000000000000 IV=0x00000000	0x4424100e192a44a441c32b043b142991
Key=0x80000000000000000000000000000000 IV=0x80000000	0x1872131b4a5fcf0af73e284f3cb5f2ff
Key=0xc27dfea157408b90eb2afe51 IV=0x4a9f7c26	0x7656f51d9542b6ae184f45996777252d
Key=0xffffffffffffffffffffffff IV=0xffffffff	0x385281713487c80c54e16d7f79ad1d27

6. Conclusion

FCSRs are an alternative to LFSRs in stream cipher design since they contain built-in nonlinearity (quadratic feedback function) that makes them resistant to correlation and algebraic attacks. As with LFSRs, FCSRs also possess statistical properties, but their representation can make the cipher more vulnerable. A new ring representation based on matrix representations has generalized the Fibonacci and Galois representations of FCSRs. This approach avoids the weakness of both representations while keeping the desirable and standard properties of FCSRs like periodicity, and high entropy. In addition, automata with a faster diffusion feature and better implementation outcomes are produced by the ring representation. In this paper, we proposed two designs of stream cipher based on Ring FCSR which produces 8-bit and 32-bit output using linear filters. We provided a highly efficient pseudorandom generator that was also simple to build, mainly in hardware but also in software. We compare keystream generation and periodicity with the existing FCSR design. It has been proven to be secure from all known attacks and has good statistical properties. In future work, these designs will be incorporated into cryptographic communication applications (devices) to measure cipher all the parameters and efficiency.

References

[1] A. Klapper and M. Goresky, "Feedback shift registers, 2-adic span, and combiners with memory," *Journal of Cryptology*, vol. 10, no. 2, pp. 111–147, Mar 1997. [Online]. Available: https://doi.org/10.1007/s001459900024

[2] A. Klapper, "Distributional properties of d-fcsr sequences," *Journal of Complexity*, vol. 20, no. 2, pp. 305–317, 2004, festschrift for Harald Niederreiter, Special Issue on Coding and Cryptography. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0885064X03001250

[3] M. Hell and T. Johansson, "Breaking the f-fcsr-h stream cipher in real time," in *Advances in Cryptology - ASIACRYPT 2008*, J. Pieprzyk, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 557–569.

[4] S. F. W, Meier, and D. Stegemann, "Equivalent representations of the f-fcsr keystream generator," in *Proc. ECRYPT Netw. Excellence, SASC Workshop*, Berlin, Heidelberg, 2008, pp. 87–94.

[5] F. Arnault, T. Berger, C. Lauradoux, M. Minier, and B. Pousse, "A new approach for fcsrs," in *Selected Areas in Cryptography*, M. J. Jacobson, V. Rijmen, and R. Safavi-Naini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 433–448.

[6] A. Klapper, "A survey of feedback with carry shift registers," in *Sequences and Their Applications - SETA 2004*, T. Helleseth, D. Sarwate, H.-Y. Song, and K. Yang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 56–71.

[7] M. Goresky and A. Klapper, "Arithmetic crosscorrelations of feedback with carry shift register sequences," *IEEE Transactions on Information Theory*, vol. 43, no. 4, pp. 1342–1345, 1997.

[8] Z. Lin, D. Lin, and D. Pei, "Practical construction of ring lfsrs and ring fcsrs with low diffusion delay for hardware cryptographic applications," *Cryptography and Communications*, vol. 9, no. 4, pp. 431–443, Jul 2017. [Online]. Available: https://doi.org/10.1007/s12095-016-0183-8

[9] M. Goresky and A. Klapper, "Fibonacci and galois mode feedback with carry shift registers," in *Proceedings. 2001 IEEE International Symposium on Information Theory (IEEE Cat. No.01CH37252)*, 2001, pp. 94–95.

[10] F. Arnault, T. P. Berger, C. Lauradoux, and M. Minier, "X-fcsr – a new software oriented stream cipher based upon fcsrs," in *Progress in Cryptology – INDOCRYPT 2007*, K. Srinathan, C. P. Rangan, and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 341–350.

[11] P. Stankovski, M. Hell, and T. Johansson, "An efficient state recovery attack on x-fcsr-256," in *Fast Software Encryption*, O. Dunkelman, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 23–37.

[12] F. Arnault, T. Berger, and A. Necer, "Feedback with carry shift registers synthesis with the euclidean algorithm," *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 910–917, 2004.

[13] R.A.Rueppel, "Analysis and design of stream ciphers," in *Springer-Verlag, Berlin Heidelberg New York*, Berlin, 1986.

[14] N. Naser and J. Naif, "A systematic review of ultra-lightweight encryption algorithms," *International Journal of Nonlinear Analysis and Applications*, vol. 13, no. 1, pp. 3825–3851, 2022. [Online]. Available: https://ijnaa.semnan.ac.ir/article_6167.html

[15] F. Arnault, T. P. Berger, and A. Necer, "A new class of stream ciphers combining lfsr and fcsr architectures," in *Progress in Cryptology — INDOCRYPT 2002*, A. Menezes and P. Sarkar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 22–33.

[16] I. M. Hayder, H. A. Younis, I. A. Abed, and H. A. Younis, "Security of enhancement of adder in stream cipher system," in *2020 1st. Information Technology To Enhance e-learning and Other Application (IT-ELA*, 2020, pp. 99–102.

[17] A. Klapper and J. Xu, "Algebraic feedback shift registers," in *Theoretical Computer Science*, vol. 226, 1999, pp. 61–93.

[18] M. Hamann, A. Moch, M. Krause, and V. Mikhalev, "The draco stream cipher: A power-efficient small-state stream cipher with full provable security against tmdto attacks," *IACR Transactions on Symmetric Cryptology*, vol. 2022, no. 2, p. 1–42, Jun. 2022. [Online]. Available: https://tosc.iacr.org/index.php/ToSC/article/view/9712

[19] A. Kesarwani, D. Roy, S. Sarkar, and W. Meier, "New cube

distinguishers on nfsr-based stream ciphers," *Designs, Codes and Cryptography*, vol. 88, no. 1, pp. 173–199, Jan 2020. [Online]. Available: https://doi.org/10.1007/s10623-019-00674-1

[20] Z. Niu and Y. Sang, "On the linear complexity of binary half-l-sequences," *International Journal of Network Security*, vol. 24, no. 3, pp. 444–449, May 2022.

[21] S. T. M, Doganaksoy, Ali, and C. Calik, "Statistical analysis of synchronous stream ciphers," in *In: SASC 2006: Stream Ciphers Revisited, Leuven*, Belgium, 2006.

[22] L. Jiao, Y. Hao, and D. Feng, "Stream cipher designs: a review," *Science China Information Sciences*, vol. 63, no. 3, p. 131101, Feb 2020. [Online]. Available: https://doi.org/10.1007/s11432-018-9929-x

[23] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, N. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," 2001-05-15 2001.

[24] F. Arnault and T. P. Berger, "F-fcsr: Design of a new class of stream ciphers," in *Fast Software Encryption*, H. Gilbert and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 83–97.

[25] T. Siegenthaler, "Cryptanalysts representation of nonlinearly filtered ml-sequences," in *Advances in Cryptology — EUROCRYPT' 85*, F. Pichler, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 103–110.

[26] G. Akula, V. Naik, and B. Umadevi, "On the generation of multiple -sequences from single wfcsr automaton for cryptographic applications," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, 2016, pp. 861–867.

[27] F. Arnault, T. P. Berger, and B. Pousse, "A matrix approach for fcsr automata," *Cryptography and Communications*, vol. 3, Jun 2011.

[28] T. P. Berger, M. Minier, and B. Pousse, "Software oriented stream ciphers based upon fcsrs in diversified mode," in *Progress in Cryptology - INDOCRYPT 2009*, B. Roy and N. Sendrier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 119–135.

[29] D. Pei, Z. Lin, and X. Zhang, "Construction of transition matrices for ternary ring feedback with carry shift registers," *IEEE Transactions on Information Theory*.

G PRAVEEN KUMAR is pursuing his Ph.D. in computer science and engineering at Department of Computer Science and Engineering, ANU College of Science, Acharya Nagarjuna University, Guntur, Andhra Pradesh, and also working as a Research Associate at CR Rao Advanced Institute of Mathematics, Statistics, and Computer Science, University of Hyderabad Campus, Hyderabad. He received his MCA degree from Osmania University. His areas of interest are cryptography, cryptanalysis, and the design of security protocols.

Appala Naidu Tentu is an Associate Professor at CR Rao Advanced Institute of Mathematics, Statistics, and Computer Science (AIMSCS), University of Hyderabad Campus. He obtained Ph.D. in Computer Science and Engineering from JNTU Hyderabad. His research interests are in the areas of cryptography, cryptanalysis, and the design of security protocols.

Neelima Guntupalli is an Assistant Professor working in the Department of Computer Science and Engineering, ANU College of Science, Acharya Nagarjuna University. She got a Ph.D. in Computer Science and Engineering (in the field of cryptography and network security). Her research areas include cryptography, cloud computing, and machine learning.

Nagendar Yerukala is presently working as Research Scientist in CRRAO AIMSCS, Hyderabad. He obtained Ph.D(CSE) from JNTUH Hyderabad. He did his M.Tech from NITK surathkal and M.Sc from Kakatiya university. His areas of interest are Network security and Cryptology.