Received: May 22, 2019

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Alternating Step Generator Using FCSR and LFSRs: A New Stream Cipher

Nagendar Yerukala[1] Venu Nalla[1] Padmavathi Guddeti[1]*
V. Kamakshi Prasad[2]

[1]*C.R. Rao Advanced Institute of Mathematics, Statistics and Computer Science, UoH Campus, Hyderabad, India*
[2]*JNTUH College of Engineering, Hyderabad, India*
* Corresponding author's Email: padmagvathi@gmail.com

Abstract: Data encryption play major role in all communications via internet, wireless and wired media. Stream ciphers are used for data encryption due to their low error propagation. This paper presents a new design of stream cipher for generating pseudorandom keystream with two LFSR`s, one FCSR and a non-linear combiner function, which is a bit oriented based on alternating step generator (ASGF). In our design two LFSR`s are controlled by the FCSR . ASGF has two stages one is initialization and the other is key stream generation. We performed NIST test suite for checking randomness on the keystream of length 1500000 generated by our design with 99% confidence level. Keystream of ASGF passes almost all NIST tests for randomness and the results are tabulated. For fixed extreme patterns of key and IV, ASGF is giving random sequence. Throughput of our proposed stream cipher ASGF is 4364 cycles/byte. Throughput comparison of ASGF with existing stream ciphers A5/1, A5/2 and RC4 are presented. It is not possible to mount brute force attack on ASGF due to large key space 2^{192}. Security analysis of ASGF against exhaustive search, Algebraic attack, distinguishing attack is also presented in this paper.

Keywords: Stream ciphers, Linear feedback shift register, Feedback with carry shift register, Randomness test, Alternating step generator, Attacks.

1. Introduction

Stream Cipher (SC) produces pseudo random keystream which is generated from a smaller secret key. Several structures for stream ciphers have been described in literature [1, 2] and they are of continuing interest. This is because of their advantage of lower computational complexity, and lack of error propagation unlike block ciphers. These however generate pseudo random sequences since after the period, the sequence repeats. These are Symmetric key ciphers since the key used for both encryption and decryption is same. These need a clock and at each end of the clock, a bit or word is generated. Further, there is a need for synchronization between the encryptor and decryptor since both need to start at the same state.

Comparing with block ciphers, stream ciphers can be implemented efficiently in Smart cards and RFID tags where we have limited hardware resources. In real world, stream ciphers are very much useful for communication which requires high throughput. In GSM communication message lengths cannot be predetermined, in which case stream ciphers work very efficiently. Synchronous stream ciphers are used due to no error propagation and better security feature. Re-sending of data packet is no major issue in comparison of security concerns due to very high speed and low error rate of present day networks. Our stream cipher design is synchronous stream cipher.

Synchronous Stream Cipher: A synchronous stream cipher with key-space: $\{0,1\}^k$ and IV -space $\{0,1\}^s$ consists of

- An ***internal state*** of n bits,
- A ***state initialization function* (SIF):** $\{0,1\}^k \times \{0,1\}^s \rightarrow \{0,1\}^n$
- A ***state update function* (SUF)**: $\{0,1\}^n \rightarrow \{0,1\}^n$, and
- An ***output function* (OF):** $\{0,1\}^n \rightarrow \{0,1\}^m$.

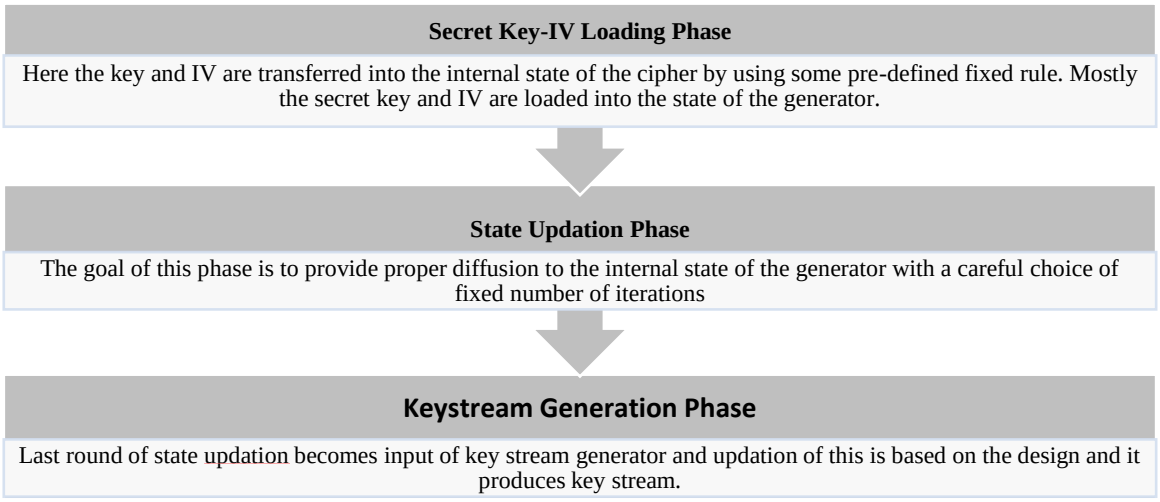

Figure. 1 Phases of stream cipher

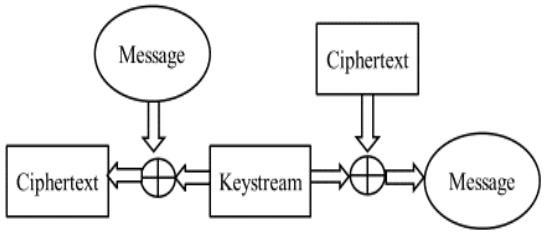

Figure. 2 General structure of stream cipher

It takes a key and an optional IV (initialization vector) pair $(K, IV) \in \{0, 1\}^k \times \{0,1\}^s$ as input and outputs a key stream $\boldsymbol{z} = (z_1; z_2, \ldots\ldots)$, *where* $z_i \in \{0,1\}^m$ is computed as follows:

Compute initial state $S_0 = \mathbf{SIF}(K, IV) \in \{0,1\}^n$ for each iteration i=1, 2,…., output $z_i = \mathbf{OF}(S_{i-1})$, and update state $S_i = \mathbf{SUF}(S_{i-1})$.

For m = 1, it is usually referred to as bit-oriented stream cipher. It is called word-based stream cipher if m > 1. Some examples are SOBER [3] and SNOW [4]. From Fig. 1, one can understand different phases involved in the cipher stream generation.

Stream ciphers use a keystream generator to generate keystream, which is added modulo-2 (XOR-ed) with the plain text message to encrypt the message. At the receiving end, the same locally generated key stream is XOR-ed with the received cipher text to obtain back the plain text message (see Fig. 2).

Traditional stream ciphers take only single input known as secret key which produces same key stream. Later to solve the problem of key management due to frequent re-keying, modern stream ciphers are fed with two inputs, one being public known as the initial vector (IV) and the second input secret key. To decrypt the message, the receiver must use the same key and IV to initialize the keystream generator thereby producing the same keystream. Keystream generator has two components: initialization and key stream generation.

Arrangement of the paper in different sections is as follows: Section 2 discussed about literature review of the proposed stream cipher (ASGF). Section 3 deals with ASGF design and its components. Section 4 deals with the analysis of ASGF. Conclusion and future work follow in Section 5.

2. Literature review

In this section we present the various previous techniques used to generate cipher streams.

2.1 Linear Feedback Shift Register (LFSR)

LFSR of length L consist of L stages of one bit memory (D Flip-Flop) and several bits are tapped from the shift register and XOR-ed to produce a bit which is fed back to the shift register. The bits to be tapped are dependent on the chosen primitive polynomial. Each stage has capacity of storing one bit. Stage-0 (s_0) is the first output of LFSR. Then content of stage-0 is moved to stage-1, that is, stage 1 moved to stage 2 and stage L-1 is the feedback bit denoted by S_L. were L is the length of the shift register (i.e. degree of the primitive polynomial). The computation of feedback bit in two successive steps is explained by the Eqs. (1), (2), and (3).

$$S_L = \sum_{(i=1 \text{ to } L)} C_i S_{L-i} = C_1 S_{L-1} + C_2 S_{L-2} + \ldots + C_L S_0 \ (mod\ 2) \quad (1)$$

$$S_{L+1}=\sum_{(i=1 \text{ to } L)} C_i S_{(L+1)-i}$$
$$= C_1S_L + C_2S_{L-1} +.....+ C_LS_1 \ (mod\ 2) \quad (2)$$

In general,

$$S_j=\sum_{(i=1 \text{ to } L)} C_i S_{j-i}$$
$$= C_1S_{j-1} + C_2S_{j-2} +.....+ C_LS_{j-L} \ (mod\ 2) \quad (3)$$

PRNG: $S_0, S_1, S_2, \ldots, S_{(2^L-1)}$,

Every sequence produced from LFSR of length L has a period 2^L-1 if the connection polynomial is primitive and of degree L. There are L shifted sequences with one cock cycle delay. Note that LFSR has to be loaded with a non-zero initial condition in the L stages since otherwise the feedback bit is zero and the state will not change with the clock.

If any 2L consecutive bits taken at any tap are given, then by using Berlekmap-Massey algorithm, one can easily find the primitive polynomial. Thus, we define Linear Complexity as the length of consecutive bits sequence required to fully define the Primitive polynomial. So LFSR alone cannot be used for pseudorandom generator/stream cipher. Several structures for increasing the linear complexity have been described in literature [1, 5]. We will consider these in the next sub-section.

2.2 Nonlinear feedback shift register (NLFSR)

NLFSR are generalizations of LFSRs which can do some operations on the contents of the LFSR which is known as non-linear filtering or shit registers can use nonlinear functions in the feedback loop while ensuing maximum period. One can combine two or more LFSRs using non-linear functions as well. Irregular clocking is also one very interesting technique since this enables not providing consecutive bits to the cryptanalyst. This increase the linear Complexity enormously. Analysis and its properties were discussed in [1, 5-8].

2.2 Feedback with carry shift register

A Feedback with Carry Shift Register (FCSR) can be seen as an alternative to an LFSR. Klapper and Goresky [9, 10] initially proposed to use FCSRs for generating pseudo-random sequences in perspective of cryptographic applications. FSCR is easily implementable. F-FCSR-H and F-FCSR are some of the FCSR based stream ciphers developed as part of the eSTREAM project [11].

The integer ***q*** for FCSR must hold the following conditions [12]:

Let $q \ \varepsilon \ Z^- \ э \ -q$ is prime and is of bit size $n+1$ where Z^- is set of negative integers and n is the size of the main register.

- Choose a ***q*** such that the order of **2** modulo ***q*** is exactly $T= (|q|-1)$ and hence the period produced by any preliminary value p such that $0<p<|q|$ is exactly T .Here both q & $T/2$ must be prime numbers
- Set $d=(1+|q|)/2= \sum_{i=0}^{k-1} d_i 2^i$

  Here k is main register length. The hamming weight $W(d)$ of the binary expansion of *"d"* is not too small. Typically, $W(d)$ is about $n/2$ or slightly greater.

  Stream ciphers designs using FCSR were discussed in [10, 12, 13]. Cryptanalysis of stream ciphers using F- FCSR were discussed in [14, 15]. Design principles of stream ciphers using FCSR presented in [16-19].

Two basic types of FCSRs are Fibonacci and Galois representations:

- In the Fibonacci representation, some feedback bits chosen based on the connection integer will decide on the new state of left most cell and the contents are shifted right by one cell position [20].
- In the Galois representation, a single feedback bit will be fed to several intermediate cell based on the connection integer [20].

Comparing Fibonacci and Galois representations, Galois is more suitable for cryptographic applications since it has quadratic transition function. Survey related to stream ciphers used in wireless communications can be found in [28]. A new stream cipher design proposed in [29] which is based on combiner generator. A chaotic based stream cipher was presented in [30], in which session key changes dynamically. Design for 5G technologies called Espresso was designed by Debrova and Hell [31]. It is based on nonlinear feedback shift register.

3. Proposed ASG using FCSR (ASGF)

The alternating step generator (ASG) presented in [21] is shown in Fig. 2. Which uses three LFSRs. The output bit of LFSR3 decides whether the LFSR1 is clocked or LFSR 2 is clocked. The output bit is XOR of outputs of LFSR1 and LFSR2. The period of the ASG is the LCM of the periods of all the LFSRs if they are chosen to be mutually prime. Some excellent and important cryptographic properties of the ASG, are (1) period (P) is very

long,(2)linear complexity (L) is high, and (3) established that almost uniform relative frequency of small output patterns on a period, [21] (of period 2^k). More precisely, If *m, n* are degrees of the primitive polynomials of two LFSRs respectively, it is proven that the period of the output sequence generated by this design is $P=P_1 .P_2 .2^k$ and $(m+n)2^{k-1}< L\leq(m+n)2^k$,

Several attacks on Gunther's ASG have been described in literature. A divide-and conquer attack by guessing the content of LFSR3 has been described [21].

In this paper, we suggest the use of FCSR in place of LFSR3. It is predictable that alike outcomes also hold if the period of FCSR is co-prime to $P_1 .P_2$.We also introduce an additional block in which we add the output bits of LFSR1 and LFSR2 with memory.

The sum $S_n= A_n \oplus B_n \oplus C_{out,n-1}$ is the output bit of the sequence whereas the carry bit is $C_{out,n}=A_n B_n+ C_{out,n-1}(A+B)$. This will destroy the correlation of the final output with the output of LFSR2 and LFSR3.

Both LFSR1 and LFSR2 have distinct primitive connection polynomials of degrees 61 and 67 respectively. Both the LFSRs have co-prime periods. Where p_1 , the period of LFSR1 is $p_1= (2^{61}-1)$ and p_2 , the period of LFSR2 is $p_2= (2^{67}-1)$. At every stage, only one LFSR is clocked and the output bit is produced in step-then-add fashion. Let the clocking/control bit of the FCSR at time t, t>=1 be denoted by C_t. If C_t=1, LFSR1 is clocked otherwise, LFSR2 is clocked. The keystream bit z_t at time t is obtained by adding the output bits of LFSR1 and LFSR2 at time t with carry.

Notations:
|| -- concatenation
⊕--The bit-wise Exclusive –OR operation
+-- Addition
∃ -- There exists
∈ -- Belongs to

3.1 Description of various components

The feedback polynomial of LFSR-1 and LFSR-2 are presented in Tables 1 and 2, respectively.

The bit length of the FCSR used in our design is 64. Thus we have chosen a negative prime q=-33364594257439900859 and the size of **q** is 65 bits. $(|q|-1)/2$ is also prime. For good cryptographic properties we have used "***q***" which is of the form 8k+3 where $k > 0$ and also ensure that the number of ones in $(q+1)/2$ is > 32. The positions of tapped bits from the FCSR register bits must not be sparse / clustered. The number of carry bits should be higher than 32(64/2). The number of carry bits in our design is 35.

Table 1. LFSR-1

Length of LFSR-1	61
Primitive polynomial	$x^{61}+ x^{40}+ x^{39}+ x^{37}+ x^{36}+ x^{35}+ x^{32}+ x^{31}+ x^{19}+ x^{17}+ x^{13}+ x^{11}+ x^9+ x^5+ x^4+ x^3+ x^2+ x+1$
Tapping bits	0, 21, 22, 24, 25, 26, 29, 30, 42, 44, 48, 50, 52, 56, 57, 58, 59, 60

Table 2. LFSR-2

Length of LFSR-2	67
Primitive polynomial	$x^{67}+ x^{35}+ x^{34}+ x^{32}+ x^{19}+ x^{18}+ x^{16}+ x^{11}+ x^{10}+ x^8+ x^7+ x^6+1$
Tapping bits	0, 32 , 33, 35, 48, 49, 51, 56, 57, 59, 60, 61

3.1.1. Combiner function

The output of our design is obtained using a full-adder:

$C_0=0$
$z_i=x_i \oplus y_i \oplus C_i \; 0 \leq I \leq n$
$C_i= x_{i-1} y_{i-1} \oplus y_{i-1} C_{i-1} \oplus x_{i-1} C_{i-1} , \; 1 \leq I < n$

Note that the period of the sequence generated is $(|q|-1) \times (2^{61}-1) \times (2^{67}-1)$.

3.1.2. Parameters

The key space and IV space are of 192-bit:
K = $\{0,1\}^{192}$
IV Space: IV= $\{0, 1\}^{64}$
Output: one bit per iteration

3.1.3. Key and IV setup

The key and IV Setup procedure should satisfy the following conditions:

- The transformation provided by the Key and IV Setup procedure should be hard to invert in order to prevent a direct key recovery attack.
- The number of iterations the generator is run before producing any output should guarantee

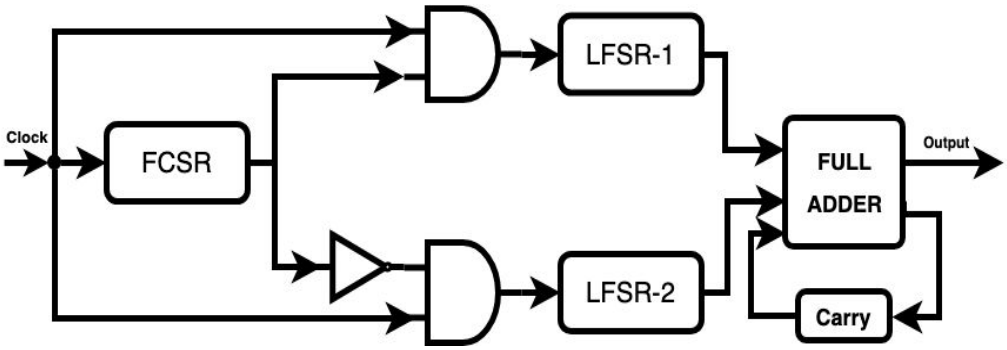

Figure. 3 Schematic of the proposed design – Alternating step generator

better diffusion in order to avoid resynchronization attacks.

Key set up/Initialization of the ASGF procedure is

- Key: $K \in \{0,1\}^{192}$ is divided into 24 bytes as $K=k_0||k_1||\ k_2||\ldots\ldots\ldots\ldots||k_{21}||\ k_{22}||k_{23}$, where each $k_i \in \{0,1\}^8$
- Initial Vector: $IV \in \{0,1\}^{64}$ is divided into eight bytes as $IV=IV_0||\ IV_1||\ IV_2||\ldots\ldots\ldots\ldots||\ IV_6||\ IV_7$, where each $IV_i \in \{0,1\}^8$

- **KEY Loading:**

a. Main register of FCSR is loaded as follows:
   i. $M[63,\ldots,0]=(k_3 \oplus IV_5)||\ k_{19}\ ||\ (k_9 \oplus IV_7)\ ||\ k_{13}\ ||\ (k_{15} \oplus IV_2)\ ||\ k_7\ ||\ (k_{21} \oplus IV_3)\ ||\ k_1$

   Carry register of FCSR is initially loaded by all zeros.

b. The two LFSRs are represented as an array and filled with the key and IV values as follows:
   i. $ARRAY[127,\ldots,0]= k_5\ ||\ (k_{20} \oplus IV_4)\ ||\ k_{11}\ ||\ k_{14}\ ||\ (k_{17} \oplus IV_1)\ ||\ k_8\ ||\ k_{23}\ ||\ k_2\ ||\ k_4\ ||\ k_{18}\ ||\ k_{10}\ ||\ (k_{12} \oplus IV_6)\ ||\ k_{16}\ ||\ k_6\ ||\ (k_{22} \oplus IV_0)\ ||\ k_0$
   ii. $LFSR1[60,\ldots,0]=ARRAY[60,\ldots,0]$
   iii. $LFSR2[66,\ldots,0]=ARRAY[127,\ldots,61]$

- If All LFSR1 and LFSR2 bits loaded are zeros, then we set LFSR1 [60] and LFSR2 [66] as 1.
- Next, iterate all two LFSRs and FCSR 70 times without clocking (Regular Clocking) and ignore the output.

3.1.4. Keystream generation

After the state initialization process, from the 71st iteration onwards, iterate the system in clock-controlled manner to yield the keystream bits, which will be XORed with plaintext bits to produce the Ciphertext.

3.1.5. Encryption/decryption:

Let the i^{th} plain text and cipher text bits be denoted as P_i and C_i.

i. **The Encryption process**
   $C_i = P_i \oplus z_i$, where z_i denotes the i^{th} keystream bit, for i = 1, 2. ...
ii. **The Decryption process**
   $P_i = C_i \oplus z_i$, where z_i denotes the i^{th} keystream bit, for i = 1, 2. . .

From Fig. 3, one can understand Schematic of the proposed design ASGF. Structure of new design ASFG is simple and can be implemented efficiently on hardware like Field Programmable Gate Array (FPGA) boards, chips, and RFID tags etc.

4. Analysis of ASGF

In this section we discussed analysis of the proposed stream cipher and possible attacks and resistance to attacks.

4.1 Exhaustive search

In an exhaustive key search, an attacker attempts all possible 2^{192} keys to determine which secret key can appropriately decrypt an encrypted stream. A brute-force attack is an inefficient form of attack since it needs trying all 2^{192} possible keys to determine which is the correct key is can take a long time. Since ASGF key size is 192, trying 2^{192} possible keys is needed for Brute force attack.

4.2 Statistical analysis

Various tests of randomness occur in the research contributions or literature [22, 23]. Golomb [24] originally suggested three randomness related properties (the balanced property, the runs property, and the autocorrelation property) known as Golomb's postulates [25]. Many wide properties of

randomness have since been established, such as DIEHARD [26], and the NIST test suite [23]. In those two, we selected to estimate the randomness of the keystream produced by the stream ciphers using the NIST SP 800-22. It consists of 15 randomness tests, which are tests (1) frequency test- to find the zeros and ones percentage and both must be closed to 0.5 (2) block frequency-frequency test within a block (3) runs –to find the number of runs or flips from 0 to 1 or 1 to 0 (4) longest run to find the longest run in the sequence (5) matrix rank- to find the some bit string has repetitive arrangements through its whole sequence by separating the string into blocks and making them into matrix to check the linear dependency (6) spectral (DFT) – to obtain the peak heights,(7)non-overlapping template- to find the too many occurrences (8) overlapping template- to find the number of occurrences,(9) a "Universal Statistical" test based on compressibility, (10) linear complexity-to find the length of LFSRs and to test the complexity of the sequence (11)serial- to find the possible pattern in the whole sequence, (12) approximate entropy- test is based on repetition of the occurrences in the sequence,(13) cumulative sums- cumulative sum of the part of the whole sequences happening in the verified sequence is too big or too small, (14) random excursions tests and (15) random excursions variant test.

Matrix rank, linear complexity, overlapping template, non-overlapping template, approximate entropy, block frequency, runs, longest run, frequency, spectral (DFT), a "Universal Statistical" test based on compressibility, serial, cumulative sums, and two random excursions tests.

We have developed application in "C" to compute the keystream sequences generated by our proposed above design. Further, we performed NIST tests on some sample sequences of length 1500000, with 99% of confidence level generated by our design. Results of these tests are presented in Table 3.

Except very few instances our sequences passed all the NIST tests. No bias of the ASG key stream sequence was found by any of 15 tests in the NIST test suite. The keystream generated by ASG is reported to have properties of random bit streams and sufficient tolerance pertinent to these attacks.

We have also studied the sequences generated for chosen keys with fixed patterns which are presented in Table 4.

Table 3. NIST test suite results of ASGF

Sl.No	Test Name	1.5M Key stream bits	
		P value	Success/Failure
1	Approximate Entrophy	0.750906	SUCCESS
2	Block Frequency	0.511781	SUCCESS
3	Cummulative Sums	0.163447 (Forward) 0.192869 (Reverse)	SUCCESS SUCCESS
4	Fast Fourier Transform	0.435839	SUCCESS
5	Frequency	0.167122	SUCCESS
6	Linear Complexity	0.022844	SUCCESS
7	Largest Run of Ones	0.380281	SUCCESS
8	Non-Overlapping Template Matchings	-	134/14
9	Overlapping Template Matchings	0.949191	SUCCESS
10	Random excursions	INSUFFICIENT NUMBER OF CYCLES	-
11	Random excursions Variant	INSUFFICIENT NUMBER OF CYCLES	-
12	Rank	0.466215	SUCCESS
13	Runs	0.505196	SUCCESS
14	Serial	0.885314, 0.467906	SUCCESS, SUCCESS
15	Universal Statistical	0.695265	SUCCESS

Table 4. Extreme patterns of key and IV inputs of ASGF

Sl.No	Parameters	Values
1	Input Keystream	0X00000000000000000000 00000000000000000000000000000000
	IV	0X0000000000000000
	Output Keystream	0x9c8d1c408f082513f0655a 3160a987d8cd39181ca5c1e1bf
2	Input Keystream	0X80000000000000000000 00000000000000000000000000000000
	IV	0X8000000000000000

	Output Keystream	0x3f5de388eb5f0bc3c885 c939806917fb6d198783b60e51bf
3	Input Keystream	0X0123456789abcdef01234567 89abcdef0123456789abcdef
	IV	0X0123456789abcdef
	Output Keystream	0xfd7f5d7a0e989342b5e3ecb50 d052c6566b467ca9b6357a5
4	Input Keystream	0Xffffffffffffffffffffffff ffffffffffffffffffffffff
	IV	0Xffffffffffffffff
	Output Keystream	0x243ee-704b8b71da2997169710d2 0ea8daf2d7ee8f7c0f2b5

In the above cases, all the NIST tests have been observed to pass.

4.3 Distinguishing attack

Distinguisher: A distinguisher takes as input a sequence of n bits and outputs whether the sequence is uniformly random or whether it the keystream sequence output by a particular stream cipher. If the probability with which the output distinguisher is correct is greater than ½ then we say that it is possible to launch a distinguishing on the stream cipher in consideration.

That is, the output keystream of a stream cipher can be statistically distinguished from a random sequence, then we can say that the cipher is not strong sufficient or weak for cryptographic applications. ASG is designed with complex initialization and update function. It has no linear covering and therefore ASG is resistant to distinguishing attacks [27].

4.4 Algebraic attack

In Algebraic Attack, cryptosystem is represented as a system of equations and then known values are replaced in the variables. Solving these over defined system of multivariate equations, we can recover secret key. These attacks exploit the fact that even if a function may have higher degree, it may have a low degree multiple.

Let x_0 be the least significant bit of the main register in the FSCR, which is also the output of FCSR.

The transition function of FCSR can be expressed by the Eqs. (4) and (5):

$$x^{t+1}_i = x^t_{i+1} + d_i \,.\, c^t_i + d_i \,.\, x^t_0 \pmod 2 \quad (4)$$

$$c^{t+1}_i = d_i\,(x^t_{i+1}.\, c^t_i + x^t_0.\, c^t_i + x^t_0.\, x^t_{i+1}) \pmod 2 \quad (5)$$

where i denotes the i^{th} cell of the register and "t" denotes the iteration number/ clock number/time.

Let the output of FCSR be x^{t-1}_0, where t >=1. Let the outputs of LFSR-1 and LFSR-2 be p^t_0 and q^t_0 respectively. Then p^t_0 and q^t_0 can be expressed by the Eqs. (6) and (7):

$$p^t_0 = p^{t-1}_0\,(x^{t-1}_0 + 1) + p^{t-1}_1 \,.\, x^{t-1}_0 \pmod 2 \quad (6)$$

$$q^t_0 = q^{t-1}_1\,(x^{t-1}_0 + 1) + q^{t-1}_0 \,.\, x^{t-1}_0 \pmod 2 \quad (7)$$

The ASG generated keystream sequence bit at time "t" is given by Eqs. (8) and (9):

$$z^t = p^t_0 + q^t_0 + carry^t \pmod 2 \quad (8)$$

$$carry^{t+1} = p^t_0 \,.\, q^t_0 + p^t_0 \,.\, carry^t + q^t_0 \,.\, carry^t \pmod 2 \quad (9)$$

where $carry^t = 0$.

The equation can have maximum degree 2(64+1) as the length of the controlling register is 64, which is incorporated into the internal states of two LFSRs.

Since length of the internal state is 192, we have to assign those many number of variables to produce system of equations. We will get nonlinear equations and solving them is NP complete. So algebraic cryptanalysis is not possible on ASGF.

4.4 Throughput analysis

We have picked some popular and standard stream ciphers for the comparison of throughput of our ASGF. This calculation has been done in two abstract parameters. The first throughput calculation is keystream generated per second by these ciphers. Secondly, first throughput result and processor's speed are used to calculate throughput in cycles of system clock.

All these implementations are done using C language, the processor Intel Core i7 CPU 860 @ 2.4 Ghz X 8, along with 8 GB RAM and 64-bit Ubuntu 14.04 LTS.

Throughput of ASGF is comparable to the established stream ciphers can be observed from Table 5.

5. Conclusion and future work

In this paper, a new design of stream cipher Alternative Step Generator using FCSR (ASGF) is proposed, which is a software oriented stream cipher to generate pseudorandom sequence with 2LFSRs

Table 5. Throughput of ASGF

Stream Cipher	Type	Key size	Throughput per sec (MB/sec)	Throughput in cycles (Cycles/ byte)	Brute force attack complexity
A5/1[33]	LFSR based	64	0.79	3038	2^{64}
A5/2[33]	LFSR based	64	0.44	5455	2^{64}
ASGF	Combination of FCSR and two LFSRs	192	0.55	4364	2^{192}
RC4[32]	byte Array based	40 to 128	14.2	169	2^{40} or 2^{128}

and one FCSR and one nonlinear combiner function with 192 bit key size and 64 bit IV. ASGF is bit oriented stream cipher. ASGF has two stages: initialisation and key stream generation. Using ASGF, we generated keystream of length 15,00,000 and tested for randomness using NIST test suites with 99% confidence level. Key stream of ASGF passes almost all NIST tests for randomness used tests. Throughput of our proposed stream cipher ASGF is 4364 cycles/byte. Throughput comparison of ASGF with A5/1, A5/2 and RC4 are presented. And brute force attack complexity of ASGF is 2^{192}. This complexity is very high when compared to popular ciphers. In ASGF, key size and combination of LFSRs, FCSR and nonlinear combiner function play a major role in resistance to several attacks. ASGF is secure enough against exhaustive search, algebraic and distinguishing attacks. For fixed extreme patterns of key and IV inputs ASGF producing completely random output. The correlations between keystream and key & keystream and IV have to be performed and exploit other attacks as future work.

Acknowledgments

The authors would like to thank Dr. P.V. Ananda Mohan, CDAC, Bangalore for his valuable suggestions. Our special thanks to anonymous reviewers for their valuable reviews.

References

[1] A.J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, ISBN. 0-8493-8523-7, 1996.

[2] W. Stallings, *Cryptography and Network Security principles and Practice*, fifth edition, Prentice Hall, 2011.

[3] P. Hawkes, and G. G. Rose, *Primitive Specification for SOBER-128*, 2003. https://eprint.iacr.org/2003/081.pdf.

[4] SNOW *Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2*, Document 2: SNOW 3G Specification, ETSI/SAGE Specification, and Version: 1.1, 2006.

[5] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Wiley, 1996.

[6] K. Zeng, C. H. Yang, and T. R. N. Rao, "Large Primes in Stream Cipher Cryptography", *AUSCRYPT-1990*, 194-205, 1990.

[7] K. Zeng, C. H. Yang, D. Y. Wei and T. R. N. Rao, "Pseudorandom Bit Generators in Stream Cipher Cryptography", *IEEE Computer*, Vol. 24, No. 2, pp. 8-17, 1991.

[8] R. A. Rueppel, Analysis *and Design of Stream Ciphers*, Springer, 1986.

[9] M. Goresky and Klapper, "Large Periods Nearly de Bruijn FCSR Sequence", In: *Proc. of Advances in Cryptology-Eurocrypt'95*, LNCS, Berlin: Springer Verlag, pp.263-273, 1995.

[10] 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, 1997.

[11] eSTREM, http://www.ecrypt.eu.org/stream/, 2008.

[12] A. Klapper and B. Preneel, "Feedback with Carry Shift Registers Over Finite Fields (extended abstract)", In: *Proc. of FSE*, Lecture Notes in Computer Science, Vol. 1008, pp.170-178, 1995.

[13] A. Klapper, "A Survey of Feedback with Carry Shift Registers", *T. SETA 2004*, Springer-Verlag Berlin Heidelberg, LNCS 3486, pp.56–71, 2005.

[14] É. Jaulmes and F. Muller, "Cryptanalysis of the F-FCSR stream cipher family", In: *Proc. of the*

12th International Conference on Selected Areas in Cryptography, Springer-Verlag Berlin, Heidelberg, pp. 20-35, 2005.

[15] F. Arnault, T.P. Berger, and C. Lauradoux, "Update on F-FCSR Stream Cipher", In: *Proc. of Network of Excellence in Cryptology (ECRYPT)*, Call for stream Cipher Primitives – Phase 2, 2006.

[16] F. Arnault, T.P. Berger, C. Lauradoux, and M. Minier, "X-FCSR – A New Software Oriented Stream Cipher Based Upon FCSRs", In: *Proc. of Progress in Cryptology (INDOCRYPT 2007)*, Lecture Notes in Computer Science, Vol. 4859, Springer, Berlin, Heidelberg, 2007.

[17] F. Arnault, T. P. Berger, C. Lauradoux, M. Minier, and B. Pousse, "A new approach for FCSRs", In: *Proc. of Select. Areas in Cryptog. SAC 2009, Revised Selected Papers*, Vol. 5867, pp. 433-448, 2009.

[18] A. Ali, "Oppel-2: A New Family of FCSR-based Stream Ciphers", In: *Proc. of IEEE International Conference on Emerging Technologies (ICET)*, pp. 75-80, 2014.

[19] A. Ali, "Feedback with carry shift registers and (in-depth) security of ciphers based on this primitive", In: *Proc. of the 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2018.

[20] M. Goresky and A. M. Klapper, "Fibonacci and Galois Representations of Feedback-With-Carry Shift Registers", *IEEE Transactions on Information Theory*, Vol. 48, No. 11, 2002.

[21] C.G. Günther, "Alternating step generators controlled by deBruijn sequences", In: *Proc. of Advances in Cryptology (EUROCRYPT '87)*, Lecture Notes in Computer Science, Vol. 304, D. Chaum and W. L. Price eds., Springer-Verlag, pp. 5-14, 1988.

[22] B.Y. Zhang and G. Gong, "Randomness properties of stream ciphers for wireless communications", In: *Proc. of the Sixth International Workshop on Signal Design and Its Applications in Communications*, 2013.

[23] NIST National Institute of Standards and Technology, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, Special publication, 2010.

[24] S. Golomb, *Shift Register Sequences*, Aegean Park Press, Laguna Hills (CA), Revised edition,1982.

[25] L. Chen and G. Gong, *Communication System Security*, Boca Raton, FL: CRC Press, 2012.

[26] G. Marsaglia, *DIEHARD Battery of Tests of Randomness* [Online], 2010.

[27] D. Coppersmith, S. Halevi, and C. Jutla, "Cryptanalysis of Stream Ciphers with Linear Masking", In: *Proc. of Advances in Cryptology — CRYPTO 2002*, Yung M. (eds). Lecture Notes in Computer Science, Vol. 2442. Springer, Berlin, Heidelberg, 2002.

[28] Y. Nagendar, V. Kamakshi Prasad, A. Appa Rao, and G. Padmavathi, "Applications of Stream ciphers in wireless communications", *International Journal of Computer Sciences and Engineering*, Vol. 6, No. 6, pp.1121-1126, 2018.

[29] N. Yerukala, G. Padmavathi, V. Nalla, and V. Kamakshi Prasad, "LFL- A New Stream Cipher for Secure Communications", In: *Proc. of IEEE International Conference on Computational Intelligence and Computing Research (IEEE ICCIC)*, 2018.

[30] A. S. Alshammari, M. I. Sobhy, and P. Lee, "Secure digital communication based on Lorenz stream cipher", In: *Proc. of the 30th IEEE International System-on-Chip Conference (SOCC)*, pp. 23-28, 2017.

[31] E. Dubrova and M. Hell, "Espresso: A stream cipher for 5g wireless communication systems", *Cryptography and Communications*, Vol. 9, No. 2, pp. 273-289, 2017.

[32] G. Paul and S. Maitra, *RC4 Stream Ciphers and its variants*, CRC Press, Taylor & Francis Group, 2012.

[33] E. Barkan, E. Biham, and N. Keller, "Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication", *Journal of Cryptology*, Vol. 21, No. 3, pp.392–429, 2008.