International Journal of Electrical and Electronics Engineers ISSN- 2321-2055 (E)
http://www.arresearchpublication.com IJEEE, Vol. No.3, Issue No. 02, july-Dec, 2011

# A NOVEL STREAM CIPHER ALGORITHM FOR DATA ENCRYPTION/DECRYPTION

**Shahnawaz Uddin**

**University Women's Polytechnic, Aligarh Muslim University, Aligarh, India**

## ABSTRACT

*The recent trends show that the demand for secure data over telecommunication in increasing day by day. The most simple and fast way of providing the data security in various secured protocols such as 3GPP, SSL, TLS, WPA2, Bluetooth, RDP, etc. is being done through stream cipher algorithms for data encryption/decryption. The security of any enciphering/encryption algorithm against any cryptanalysis attack depends on the quality of a robust and random key generation method used during the encryption of data. In this research work, a novel way of random number generation using linear feedback shift registers (LFSRs) incorporating a nonlinear function has been proposed & simulated in MATLAB. It generates an output sequence of 255 random bits per session of data encryption with the help of two shift registers initialized by the secret key bits as initial vector. Due to the simplicity (less hardware) and quick response time, this algorithm can be implemented in high speed or real time communication links with very good quality of randomness.*

*Keywords:****Stream Cipher, Encryption, Decryption, Data Security, Key Generation, Linear Feedback Shift Register, Cryptanalysis Attacks.***

## I. INTRODUCTION

The most practical technique to secure the private stored/transmitted data against any unauthorized access is cryptography. To protect any data, the symmetric cryptographic techniques may encipher the data into either block cipher form or stream cipher form. To meet the demand for data security on high speed communication networks or fast storage devices, the speed of data encryption algorithms must also be increase [1-4]. Stream cipher algorithms are ideal choice for the data encryption/decryption in high speed communication/information processing applications because they are generally much faster than the block ciphers since the binary keystream used for encryption/decryption is generated independently and prior to encryption/decryption. The stream ciphers are symmetric key ciphers in which a binary keystream (a sequence of pseudorandom bits) is combined with plaintext bits to generate encrypted data and same keystream is used to decrypt the encrypted data at the receiver side. The pseudorandom (PN) sequence generator is the integral part of stream ciphers in which the binary keystream is typically generated serially using a random initial/seed value (i.e., a cryptographic key) using digital linear feedback shift registers (LFSRs). The same initial/seed value serves as the cryptographic key for generating the binary keystream for the decryption of the enciphered data. The stream ciphers have various attractive features in terms of security, fast speed, and error propagation etc. The security of the data transmission or storage in public domain is supported by various well known secure protocols like 3GPP, SSL, TLS, WPA2, etc. to fight against many attacks used to decrypt them. These protocols incorporates various stream cipher algorithms [5-11]. The strength of stream ciphers to fight against unauthorized access to stored/transmitted secret data depends on the method of random key generation but it is practically impossible to implement a random key generator to generate a true random number sequence. Therefore, the best way to

combat the problem of decryption of the private data by unauthorized entity is to generate lengthy PN number as a key in a nonlinear manner because linear PN generators are not strong enough against algebraic/correlation attacks. To generate the strong stream ciphers, we use the secret key as an internal state and an initial vector (IV) to update the operation of each round of process of pseudorandom (PN) sequence and the Boolean functions used in PN sequence generation must be good non-linear type [12].

In this research paper, an efficient stream cipher algorithm designed in such a way that can generate 255 pseudorandom bits in one round/session of process. The feedback to LFSRs generating PN-sequence is selected on the basis of the status of the of another smaller PN-sequence generator which increases the security of the stream ciphers by keeping the system simple from the point of view of design and hardware/software implementation. The 255 pseudorandom bits generated by this algorithm implemented in MATLAB per session of processing are capable to resist algebraic and correlation attacks.

## II. BACKGROUND

The individual bits are encrypted in stream ciphers by adding a bit from a secret key stream to a bit from plaintext in a synchronous or an asynchronous manner. The stream ciphers where the key stream depends only on the key, and asynchronous ones where the key stream also depends on the ciphertext. If the dotted line in figure (1) is present, the stream cipher is an asynchronous one. Most practical stream ciphers are synchronous ones. Because stream ciphers tend to be small and fast, they are particularly relevant for applications with little computational resources, e.g., for cell phones or other small embedded devices. However, stream ciphers are sometimes also used for encrypting Internet traffic. Traditionally, it was assumed that stream ciphers tended to encrypt more efficiently than block ciphers. Efficient for software-optimized stream ciphers means that they need fewer processor instructions (or processor cycles) to encrypt one bit of plaintext. For hardware-optimized stream ciphers, efficient means they need fewer gates (or smaller chip area) than a block cipher for encrypting at the same data rate.

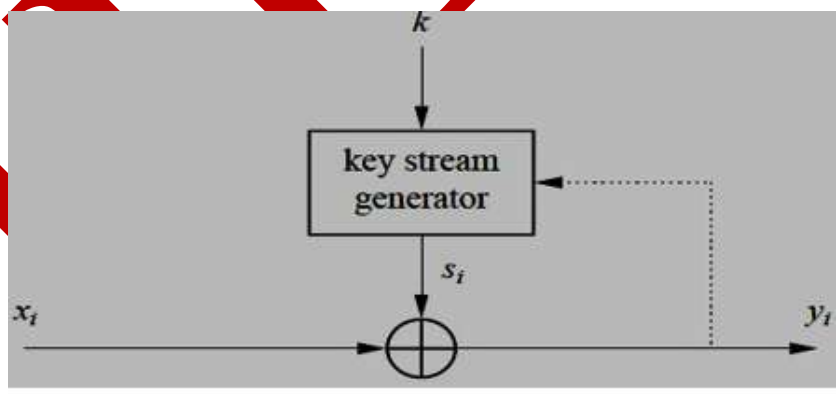



**Figure (1)** Synchronous and Asynchronous Stream Ciphers

### 2.1 Random Number Generators

From the security point of view, the most important and central issue in the stream cipher algorithms is the age band of key stream ($s_i$). Therefore, the algorithm for generation of the key stream must be the most secured, random and robust one so that the key stream bits generated should appear like a random sequence to any attacker. The purpose of the random sequence generators (which may be a software/hardware/a combination of software and hardware) to generate random numbers that behave as if they were produced randomly from the

outcome of a random event but the practical random numbers generated in the field of computing are not true random numbers (they are pseudorandom numbers) [13-14]. As random key stream plays an important role, different kinds of random number generators (RNG) will be discussed first in the following paragraphs.

**(i) True Random Number Generators (TRNGs)**have the characteristic that their output can't be regenerated, like the recording the outcomes of tossing a coin 200 times in the form of a sequence of 200 bits. The probability of reproducing the same sequence of 200 bit is extremely very small (i.e., $1/2^{200}$), and virtually it will eliminate any chance of recreating the same bit sequence. Some of other physical events of generating true random numbers are semiconductor noise, rolling of a dice, radioactive decay and clock jitter in digital circuits. For generating the secret session keys that are distributed among the authentic parties, we often need TRNGs.

**(ii) Pseudorandom Number Generators (PRNGs)**produce pseudorandom numbers using an initial seed value ($s_0$) and computed recursively in a following manner:

$s_0$= initial seed value

$s_{i+1}= f(s_i, s_{i-1}, \ldots, s_{i-t})$, i= 0,1,2, . . . and t is a fixed integer.

As we know that PRNGs not being random in true sense, but they should have the best possible statistical properties, i.e., the numbers thus generated should behave like random numbers. They are easy to generate for all practical works in the field of cryptography, for example, the rand() function in ANSI C generates pseudorandom integer numbers.

**(iii) Cryptographically Secure Pseudorandom Number Generators (CSPRNGs)** generate the pseudorandom numbers with an additional property of unpredictability, meaning that from a given n-bits of the key stream ($s_i, s_{i+1}, \ldots, s_{i+n-1}$), it is computationally very difficult to predict the next sequence of bits ($s_{i+n}, s_{i+n+1}, \ldots$). In other words, there no known algorithm which can predict next bit ($s_{n+1}$) from the knowledge of n-consecutive bits of the key sequence better than 50% probability.

**2.2 Stream Ciphers**

**2.2.1 One Time Pad (OTP)** is a stream cipher in whichthe key sequence bits($s_0, s_1, s_2, \ldots$) areproduced using a true random numbergenerator, the key bit sequence is known to the legitimateparties, andevery key bit stream ($s_i$) is used only once. The OTP stream cipher is computationally secure not unconditionally secure or non-breakable.A cryptosystem is said to be unconditionally secure if it can't be decrypted using infinite number of computational resources, i.e., the security of the system is not limited by the computational power of the attackers. For instance, a gedankenexperiment: assume a symmetric algorithm for encryption/decryption having a key stream length of 10,000 bits, and the only attack is a brute-force attack (i.e., an exhaustive key stream search). Since an attacking party may have infinite number of computational resources, say $2^{10000}$ number of computers are available and every computer checks for exactly one key. These computations will provide us the correct key stream in one step. However, of course, it will would not be feasible for the humankind to have $2^{10000}$ number of computers. We have to note down here that the situation would be different if the key bit stream values ($s_i$) are not true random numbers. Therefore, in practice it is very difficult to fulfil the requirements of an OTP cryptosystem: (i) a need for TRNG (e.g., a device based on semiconductor noise), (ii) a need for secretly delivering the true random key sequence between two legitimate parties (e.g., a trusted courier service) and (iii) need for a fresh and long key stream per session meaning a new key bit for every plaintext bit-

a major drawback of the OTP. Thus, for all these difficulties, OTPs are rarely used in practical cryptosystems but they provide us the best possible idea for the computationally secure ciphers [14].

### 2.2.2 Practical Stream Ciphers

As we know that OTPs are almost unconditionally non-breakable but they have practical difficulties to develop them. In practical stream ciphers, we replace the true random number generator by a pseudorandom number generator where the secret key bit sequence (k) serves as an initial vector as shown in figure (2). Unlike OTPs, practical stream ciphers are not computationally non-breakable, but we hope for the best possible practical computational security which is defined as "a cryptosystem is said to provide practical computationally security if t operations are required to decrypt it by the best possible known algorithm". Therefore, what a designer can assure that the designed crypto schemes are computationally secure. The design of figure (2) has the major merit OTP as the legitimate parties (Alice & Bob) need only to share a secret key stream that is a few100 bits long which is used as initial feed for generating the long pseudorandom sequences for encrypting/decrypting the messages as long we want. Now, we have only to study carefully about the characteristics of the pseudorandom sequence generators generating the bit stream ($s_i$) which are discussed as following:

**(i) Building Key Streams from PRNGs:** Some of the PRNGs show very good statistical properties that are required for a strong stream cipher, e.g., a statistical test may show that the generated bit sequence is the outcomes of tossing a coin (a random event). Thus, we are tempted to assume that the PRNGs may be employed to generate the random bit sequences but this can't be a sufficient criteria for a strong stream cipher as our attacking party is smart one which can compute the secret key stream by knowing a few pieces of plaintext and corresponding ciphertext and then can decrypt the complete ciphertext.

**(ii) Building Key Streams from CSPRNGs:** Here we employ a CSPRNG to make the key stream unpredictable and to avoid the kind of attack on the ciphertext. In CSPRNG, it is not computationally feasible to generate the bit sequence ($s_{n+1}, s_{n+2}, \ldots$) from the knowledge of the first n-bits of the key stream ($s_1, s_2, \ldots, s_n$). As we know that all PNRGs used outside of cryptographic applications are cryptographically insecure. Therefore, in practice, we require to use specially designed secured PNRGs for stream ciphers.

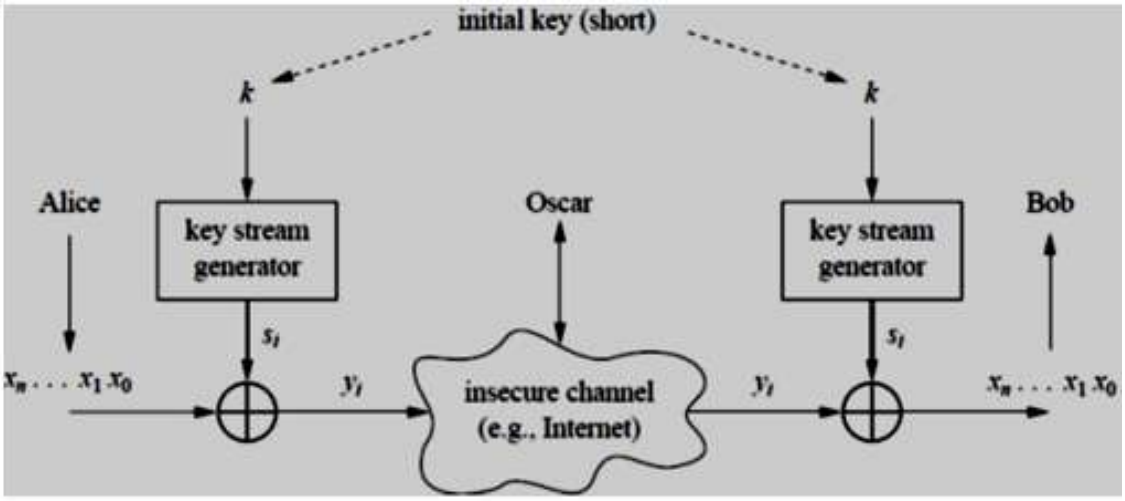



Figure (2) Practical Stream Ciphers

Now question arises that how to design a practical stream cipher algorithm that is optimized either from the software implementation point of view (that requires fewer CPU instructions to generate one key sequence bit) or the hardware implementation point of view (that requires simple design circuitry/operations to be realized in

hardware easily). The most popular and simple design example is shift registers with feedback to be discussed next.

**2.3 Linear Feedback Shift Registers (LFSRs)**

The practical stream ciphers employ the key stream generators to generate a sequence of key bits ($s_1$, $s_2$, …) having certain good statistical properties. A very simple and practical way of realizing a long pseudorandom bit sequences is to use LFSRs. An LFSR is nothing but a series of clocked cascaded storage elements (flip-flops or FFs) with a feedback path. The degree of LFSRs is given by the number of FFs used in its design (e.g., an LFSR with m-FFs is having a degree m). The feedback network circuitry computes the input to the last FF as modulo-2 addition (XOR-sum) of the outputs from the particular FFs in the shift register. Figure (3) shows a general hardware implementation of an LFSR with degree m which employs m-FFs and possible m-feedback paths in combination with XOR operation. The values of feedback coefficients ($p_0$, $p_1$, … , $p_{m-1}$) decide the activeness of a feedback path, e.g., $p_i$= 1 means the feedback is active (closed switch) and $p_i$= 0 means the respective FF output is inactive for the feedback (open switch). Thus, we can obtain a mathematical expression for the feedback path using this notation, i.e., multiply the output of $i^{th}$ FF ($s_i$)by its coefficient value ($p_i$) which results into either the output value $s_i$ (if $p_i$= 1) or the output value 0 (if $p_i$=0). Therefore, the feedback coefficients play a very crucial role for the output bit sequence generated by the LFSR. For example, suppose an LFSR is loaded initially with the initial vector values ($s_0$, $s_1$, … , $s_{m-1}$), the next value of the output bit of the LFSR ($s_m$), that is also the input to the last FF (leftmost FF) can be generated by taking the modulo 2 addition (XOR-sum) of the products of FF outputs ($s_i$) and respective feedback coefficients ($p_i$):

$s_m \equiv s_{m-1}\, p_{m-1} + \ldots + s_1\, p_1 + s_0\, p_0 \bmod 2$

Similarly, the next LFSR output bit can be computed as:

$s_{m+1} \equiv s_m p_{m-1} + \ldots + s_2\, p_1 + s_1\, p_0 \bmod 2$

And in general, the output bit sequence can be described by the following equation (1).

$$\mathbf{s_{i+m} = \sum_{j=0}^{m-1} p_j\, s_{i+j} \bmod 2;\quad s_i, p_j \in \{0, 1\}; i = 0, 1, 2, \ldots} \qquad (1)$$

Thus, sometimes the LFRS are known as linear recurrences because the next output bit values are generated through a combination of some previous output bit values. And the output bit sequence generated by an LFSR repeats its values periodically because the recurring states are finite. Moreover, the generated output bit sequences from an LFSR with degree m have different lengths ($\leq 2^m$ -1) that depend on the feedback coefficients and the maximum sequence length generated by an LFSR is $2^m$ -1. The next state of an LFSR is uniquely computed from the previous status of m-internal register bits. And an LFSR is said to start repeating its values as soon as it assumes its initial state. As an m-bit vector can only have ($2^m$ -1) nonzero states, thus the maximum sequence length is ($2^m$ -1) before repetition. We exclude all zero state because if an LFSR assumes all zero state then it will get "caught" into it and will never be able to come out of it again. Note that only some particular configurations of feedback coefficients ($p_0$, $p_1$, … , $p_{m-1}$) result maximum length sequence from LFSRs. The LFSRs with a feedback coefficients ($p_{m-1}$, … , $p_1$, $p_0$) are usually represented by the following polynomial notation: $P(x) = x^m + p_{m-1}x^{m-1} + \ldots + p_1 x + p_0$. For example, the LFSR with feedback coefficients ($p_3$ = 0, $p_2$ = 0, $p_1$ = 1, $p_0$ = 1) may be represented by the polynomial $P(x) = x^4 + x + 1$. An alternative notation (5, 2, 0) represents the polynomial $P(x) = x^5 + x^2 + 1$. The maximal length LFSRs are also have primitive polynomials (a special type of irreducible polynomials meaning that they only divided by 1 and polynomial itself). Note that

there are many primitive polynomials for every given degree m that can easily find out. For example, an LFSR of degree m = 4 with the feedback path coefficients ($p_3$ = 0, $p_2$ = 0, $p_1$ =1, $p_0$ = 1) generates the output bit sequence with a period of $2^m$ -1 =$2^4$ -1= 15, i.e., it is a maximum-length LFSR. But in case of an LFSR of degree m=4 and feedback coefficients ($p_3$ =1, $p_2$ =1, $p_1$ =1, $p_0$ =1), the generated output bit sequence has period of 5; thus, it is not a maximum-length LFSR [14].

LFSRs can be easily realized in software/hardware to produce a long bit stream with good statistical properties but not strong enough cryptographically, means, they can be easily decrypted. Therefore, we have to introduce some kind of non-linearity in the design of PN-sequence generator to make it difficult to decrypt the ciphertext as much as possible.

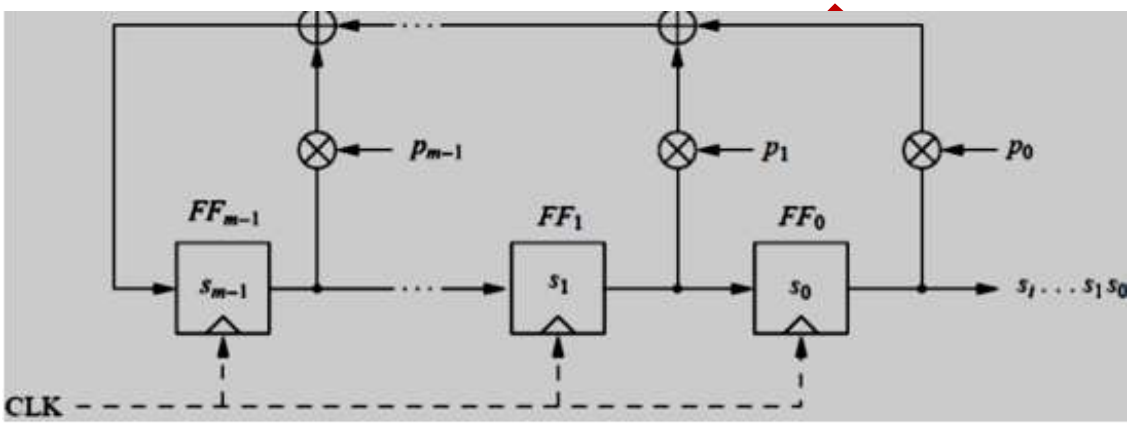



Figure (3) General LFSR with Feedback Coefficients ($p_i$) and Initial Values ($s_{m-1}$, …, $s_1$, $s_0$)

## III. NOVEL STREAM CIPHER ALGORITHM

Here, we exploited the simplicity of implementing the LFSRs in software as well as in hardware. But to make the cryptosystem computationally secured, a non-linearity has been introduced in the design structure of the PN-sequence generator to avoid the algebraic and correlation attacks. Usually, the strong random generators are structured by combining multiple LFSRs that may work together with variations in method to provide non-linearity in output bit stream. So, here we designed and simulated a pseudorandom number generator (PRNG) with novel LFSR substructure with very good random statistics that may have useful applications in symmetric stream ciphers. As shown in figure (4), this PN-sequence generation scheme is different from the conventional one. It has employed two separate maximal PN-sequence generators (a primary with feedback coefficients (1,2,3,4,6,8) and a secondary with feedback coefficients (1,4)), where the feedback coefficients/connections of the primary PN-sequence generator are selected/deselected by the status of the secondary PN-sequence generator. Here, the feedback connections/taps of the main PN-sequence generator are varied continuously by means of the status of the secondary PN-sequence generator with degree m=4 which is generating 15-bits maximal PN-sequence for experimental study. The four output status of the secondary LFSR is used to control the feedback coefficients of the primary PN-sequence generator by feeding them through AND gates whose outputs fed to the XOR gates. If the status of the particular output bit of the secondary LFSR is 1, it means that the corresponding feedback coefficient is selected in generating the main PN-sequence otherwise not. The feedback coefficients/connections (1,8) of primary PN-sequence generator are fixed and others (2,3,4,6) are controlled by the status of the secondary LFSR. The clock of the secondary PN-generator is taken after dividing the original clock by an integer number (N) so that the feedback connections of the main PN-generator remain

same for the N-bits of the output bit stream generated and after N-bits the feedback connections may change. In this way, the randomness of the PN-sequence generator increases by selecting/deselecting the feedback coefficients and varying the values of N. This novel algorithm is used for generating the symmetric stream ciphers with the 12-bits as initial vector for the initialization purpose of both the PN-sequence generators (8-bits for primary one and 4-bits for secondary one). This 12-bits sequence is kept secret and form the part of the symmetric secret key.

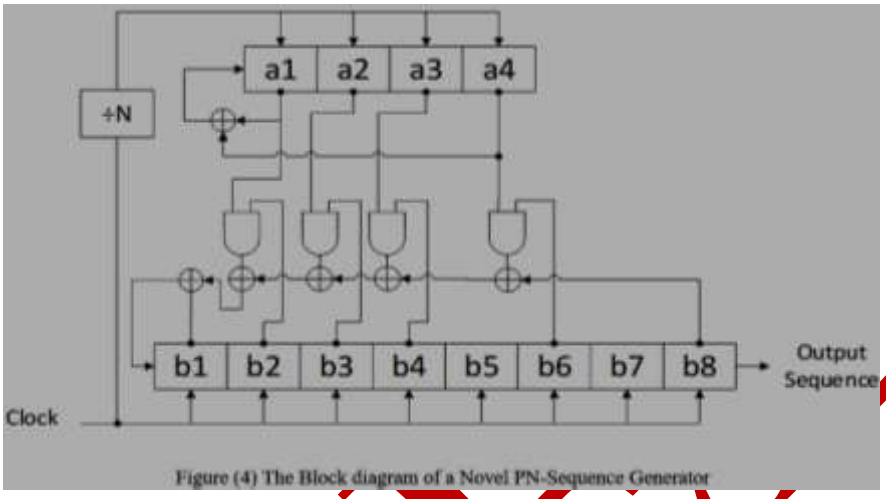
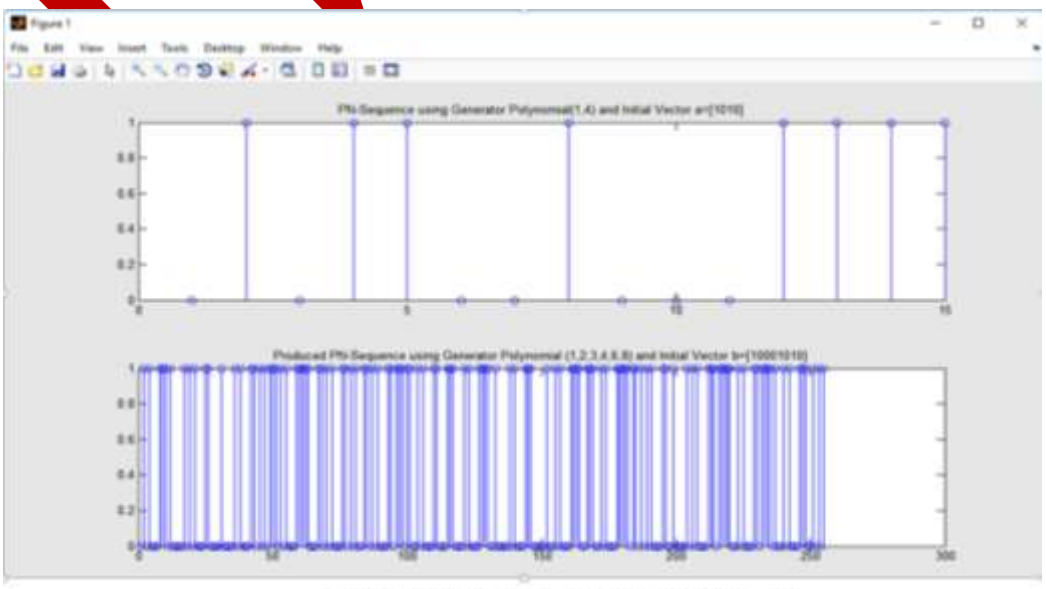


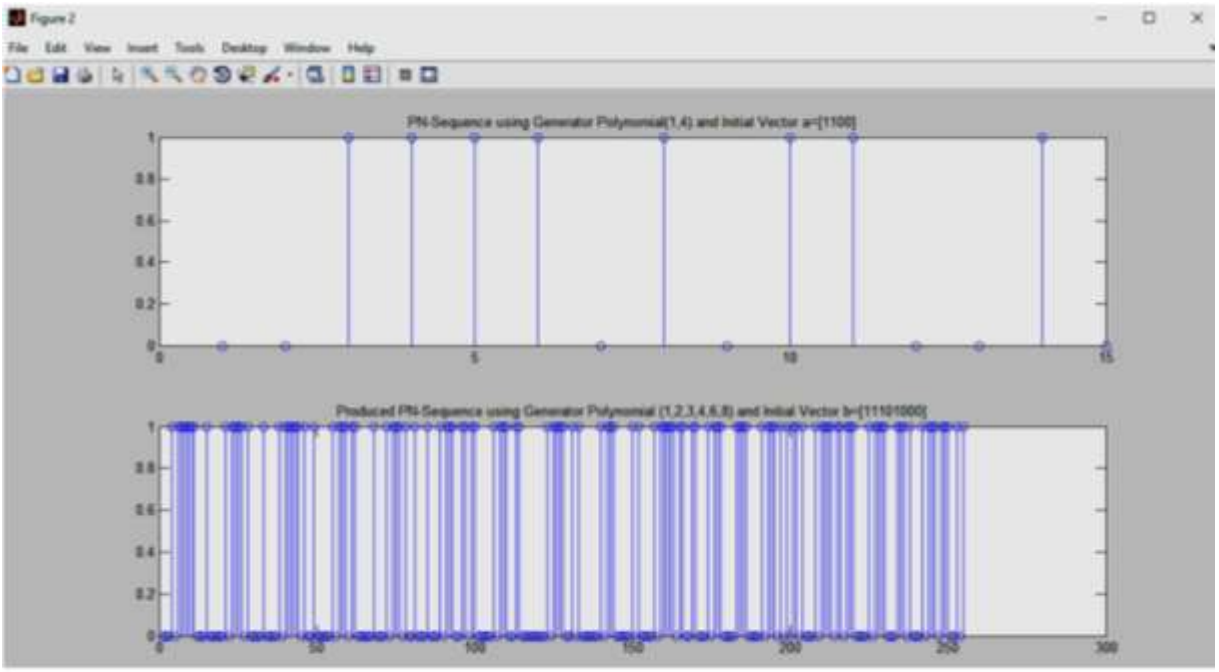

Figure (4) The Block diagram of a Novel PN-Sequence Generator

## IV. RESULTS & DISCUSSION

The proposed novel algorithm for generation of PN-sequence in MATLAB on Windows platform. The results obtained are presented in figure (5) (a-b) for different initial vectors for both PN-sequence generators, and it is clear that the scheme presented very satisfactorily results in randomness. The secondary PN-sequence generator generates 15-bits maximal sequence and first 255-bits of primary PN-generator have been shown in the results. The output bit sequence of this novel PN-sequence generator depends on the initial vectors of the both PN-sequence generators per session/round of the encryption/decryption process. If the initial vectors/conditions of the shift registers change, the output bit sequence also changes. Here, N is taken 16 for the simulation purpose, however, simulation studies have also been done for N equal to 4 & 1 with the expected results.



(a) For Initial Vectors a=[1010] & b=[10001010]




(b) For Initial Vectors a=[1100] & b=[11101000]

Figure (5) Generated Output PN-Sequence for N=16

## V. CONCLUSION

In this research paper, some weaknesses and problems associated with well-known PN-sequence algorithms have been discussed that are threatening in different applications of cryptography. This new algorithm has been designed based on 2 PN-sequence generators, where one PN-sequence generator is employed to select/deselect the feedback coefficients/connections to make the design more robust against the algebraic and correlation attacks. As the circuitry used in the block diagram is having fewer blocks and they all are easily available, it can also be implemented in hardware without any problem. As the speed of processing is one of the important parameters, therefore, it is possible to implement designed PRNG by software and hardware to obtain suitable speed of processing by using clock of high frequency. This research presented a novel methodology to generate a long unique output bit sequence of bits using as a key of stream cipher system. For the experimental purpose, two shift registers (4-bits and 8-bits) in serial in parallel out (SIPO) mode have been employed. But larger sizes of the shift registers may also be used for generating long PN-sequence for industrial purpose.

## REFERENCES

[1] K.C. Zeng, C. H. Yang, and T.R.N.Rao, “Pseudorandom Bit Generators in Stream-Cipher Cryptography”, IEEE Computer, 24, 2(1991), pp 8-17

[2] Hoonjae Lee and Sangjae Moon, “Parallel stream cipher for secure high-speed communications”, Signal Processing 82 (2002) 259 – 265, Elsevier Science B.V.

[3] A. J. Menezes, P. C. Oorschot, S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL, 1997

[4] B. Schneier, “Applied Cryptography”, 2nd Edition (2007), Wiley, ISBN-10: 8126513683, ISBN-13: 978-8126513680

[5] Matt J. B. Robshaw, “Stream Ciphers, RSA Laboratories Technical Report”, TR-701 Version 2.0, July 1995

[6]Christof Paar and Jan Pelzl, "Understanding Cryptography", Springer, 2009

[7] Briceno, M., I. Goldberg, and D. Wagner, A pedagogical implementation of A5/1. URL: http://www.scard.org/gsm/a51.html.

[8] Maximov, A., T. Johansson, and S. Babbage, "An Improved Correlation Attack on A5/1, in Selected Areas in Cryptography", H. Handschuh and M. Hasan, Editors. 2005, Springer Berlin / Heidelberg. p. 1-18.

[9] Stubblefield, A., J. Ioannidis, and A. Rubin, A key recovery attack on the 802.11 b wired equivalent privacy protocol (WEP). ACM transactions on information and system security (TISSEC), 2004. **7**(2): p. 319-332.

[10] Golić, J., V. Bagini, and G. Morgari, Linear Cryptanalysis of Bluetooth Stream Cipher, in Advances in Cryptology — EUROCRYPT 2002, L. Knudsen, Editor. 2002, Springer Berlin / Heidelberg. p. 238-255.

[11] Majid Bakhtiari and MohdAizainiMaarof, "An Efficient Stream Cipher Algorithm for Data Encryption", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011, ISSN: 1694-0814

[12] Meier, W. and O. Staffelbach, Nonlinearity Criteria for Cryptographic Functions, in Advances in Cryptology — EUROCRYPT '89, J.-J. Quisquater and J. Vandewalle, Editors. 1990, Springer Berlin /Heidelberg. p. 549-562.

[13] W.W Tsang C.W. Tso Lucas Hui K.H. Pun C.F. Chong H.W. Chan "Development of Cryptographic Random Number Generator", Department of Computer Science and Information Systems, The University of Hong Kong, August 2003

[14] C. Paar and J. Pelzl, "Understanding Cryptography", DOI 10.1007/978-3-642-04101-3_2, Springer-Verlag Berlin Heidelberg, 2010