International Journal of Computer and Communication Technology

Volume 3 | Issue 2 Article 11

April 2012

NON LINEAR FEEDBACK STREAM CIPHER

.R. Siva Ram Prasad
Research Director, Dept. of CSE, Head, Dept. of CSE, Head, Dept. of CSE, Acharya Nagarjuna University, R.K College of Engineering, Narasaraopeta Engineering College Guntur – 522510, A.P., India, raminenisivaram@yahoo.co.in

G. Murali
Head, Dept. of CSE, R.K College of Engineering, Vijayawada.- 521456, A.P., India., muralig521@gmail.com

S. Gopi Krishna
Head, Dept. of CSE, Narasaraopeta Engineering College, Guntur Dt.- 522601, A.P., India, gks24@rediffmail.com

NON LINEAR FEEDBACK STREAM CIPHER

*Dr .R. Siva Ram Prasad
Research Director, Dept. of CSE,
Acharya Nagarjuna University,
Guntur – 522510, A.P., India.
Email: raminenisivaram@yahoo.co.in

**G.Murali
Head, Dept. of CSE,
R.K College of Engineering,
Vijayawada.- 521456, A.P., India.
Email: muralig521@gmail.com

***S.Gopi Krishna
Head, Dept. of CSE,
Narasaraopeta Engineering College
Guntur Dt.- 522601, A.P., India
Email: gks24@rediffmail.com.

Abstract-- **The main aim of this paper is to develop a new generation and innovative security software for applications. We proposed new stream cipher called NLFS. NLFS means Non-linear feedback stream cipher, which is a fast and secure stream cipher for e-governance applications. This stream cipher uses AES secure non-linear function and AES key generation. NLFS uses *primitive polynomial generated S-box*es in byte substitution step. NLFS uses two similar AES round functions and these two proceed parallelly to produce key-stream. *Non-linear *function* of NLFS has AES non-linear function steps (add-round key, byte substitution, mix column, shift rows) and it extra includes value-based rotation step. In value based rotation step it rotates each 8-bit word by its first 3-bit (decimal) value.NLFS have two modes basic mode that is synchronous mode and self synchronous mode. In synchronous mode key stream is independent of plain text and cipher text. In self-synchronous mode key stream generation depending on cipher text. In self synchronous mode generated key-stream update first 512-bit buffer and cipher text update the second buffer.**

Keywords: NLFS, e-governance, stream cipher, add-round key, byte substitution, mix column, shift rows

1. INTRODUCTION TO THE NLFS STREAM CIPHER

After information revolution several government, NGOs, and private corporate have been initializing several e-governance projects for their respective problems, even though security threat became a posing problem to the e-governance project across the globe. In this respect, there is a need to develop a security algorithm for providing absolute security to e-governance projects. A stream cipher is a type of symmetric encryption algorithm. Stream ciphers can be designed to be exceptionally fast, much faster than any block cipher. While block ciphers operate on large blocks of data, stream ciphers typically operate on smaller units of plaintext, usually bits. A stream cipher generates *key stream.* Encryption is accomplished by combining the key stream with the plaintext, usually with the bitwise XOR operation. The generation of the key stream can be independent of the plaintext and cipher text, yielding what is termed a *synchronous* stream cipher, or it can depend on the data and its encryption, in which case the stream cipher is said to be *self synchronizing*. Most stream cipher designs are for synchronous stream ciphers.

Block ciphers take more time to encryption or decryption of user's message. They are very secure but fail to meet user's fast requirement. Stream ciphers were developed to overcome above problem and earlier stream ciphers compromise at security level. Later designed stream ciphers provide good security by using some non-linear functions. Block ciphers need higher hardware requirements for encryption and decryption. They are not suitable for small devices who are having limited memory and hardware units, example cell phones; they have low memory and low processing units. Generally Stream ciphers are developed by using less hardware requirements. In general, stream ciphers consist of three standard pieces: an internal state, a next-state function and transformation function which converts plaintext to cipher text. The internal state is generally small, maybe a hundred bits and can be thought of as the key. The next-state function updates the state.

The transformation function takes a piece of plaintext i.e. plaint text block, it mixes with the current state, and produces the same size cipher text. The security of this scheme is based on how cryptographically annoying the two functions are. Sometimes just one of the functions is cryptographically annoying. In electronic stream ciphers, a complicated next state function is usually combined with a simple transformation that takes the low-order bit of the state and XORs it with the plaintext. In rotor machines, such as the German Enigma, the next-state function was a simple stepping of various rotors, and the transformation function was very complicated. Sometimes both are cryptographically complicated.

Traditionally, stream cipher algorithms were implemented in hardware, and needed as few gates as possible. They had to be fast. The result was many designs based on simple mathematical functions: e.g., linear feedback shift registers (LFSRs). Recently, new stream ciphers were designed for computers and not

for discrete hardware. Instead of producing output a bit at a time, they produced output a byte at a time (like RC4), or 32 bits at a time.

2. LIMITATIONS OF PREVIOUS STREAM CIPHERS

Stream ciphers essential requirements of stream cipher are fastness, less hardware implementation cost and power consumption; it is compatible to run on different platforms 32-bit, 64-bit or 8-bit processors and security. Stream ciphers for software applications with high throughput requirements and stream ciphers for hardware applications with restricted resources such as limited storage, gate count. The main evaluation criteria for stream cipher likely to be long term security, efficiency, flexibility and it needs to meet outside world requirements.

1. Security is the most important criteria since it is achieve confidence
2. The performance of the stream cipher in the specified environment is important. For software, the range of environments may include 8-bit processors, 32-bit processors to the modern 64-bit processors. For hardware, both FPGAs and ASICs may be considered.
3. Different stream ciphers will perform best against different profiles
4. Stream cipher is simple to use for ordinary users who is not expert in security field.

Already developed stream ciphers SCREAM, PY, SNOW, SOSEMANUK and GRAIN are very fast, they take only 2 to 4 clock cycles for a single byte. The above stream ciphers fail to meet security requirement (broken). SCREAM suffer from linear distinguish attack and it uses approximately O (2^{100}) output words. PY cipher is suffer from distinguish attack on key stream which requires around O (2^{72}) bytes of output and comparable time. SNOW stream cipher suffers from guess and determines attack, which as a data complexity O (2^{4}) and process complexity O (2^{224}) and key size is 256 bits. SOSEMANUK suffer from guess and determine attack, which requires O (2^{224}) computations and its key length 256. GRAIN stream cipher suffer from linear distinguish attack with time complexity O (2^{54}) when O (2^{51}) bits of key stream available.

3. MOTIVATION OF NLFS

Above mentioned stream ciphers (SCREAM, PY, SNOW, SOSEMANUK and GRAIN) are fast but broken. Above ciphers use non-linear functions in their design, but those nonlinear functions fail to prevent various attacks. By using secure non-linear function, we can prevent attacks on stream ciphers. Secure AES non-linear function prevents all attacks on AES block cipher except side channel attack. We can use AES non-linear function in stream ciphers to meet the security requirement of stream ciphers. For software applications different stream ciphers were developed and for hardware applications different stream ciphers were developed. We can use same stream cipher for software applications as well as hardware applications.

The ECRYPT NOE (European Network of Excellence for Cryptology) collect the software application stream ciphers in profile 1and hardware application stream ciphers in profile 2. That organization divides these all stream ciphers into 3 phases and each phase having profile 1 and profile 2. Phase 1 contains proposed stream ciphers till March 2006. Phase 2 contains all stream ciphers in phase 1 except broken stream ciphers and it includes proposed stream ciphers from April 2006 to March 2007. Phase 3 includes all stream ciphers in phase 2 except broken stream ciphers and it includes proposed stream ciphers from April 2007

4. DESIGN OF NLFS STREAM CIPHER

NLFS (Non-linear feedback stream cipher) is a fast and secure stream cipher for software applications. This stream cipher uses AES secure non-linear function and AES key generation. NLFS uses *primitive polynomial generated S-box*es in byte substitution step. NLFS uses two similar AES round functions and these two proceed parallelly to produce key-stream. *Non-linear function* of NLFS have AES non-linear function steps add-round key, byte substitution, mix column, shift rows and it extra includes value based rotation. (Byte substitution stage distinguished these two non-linear functions).

NLFS takes 256-bit Initial vector and 128-bit key. NLFS contain huge internal state 1024-bits and these 1024-bit internal state divide into two 512-bit buffers A and B (refer to figure 13) both are non-linear buffers i.e. both are updated by non-linear functions output. Internal state initialized by secret key and initial vector (IV). The outputs of both non-linear functions are XOR ed and generates 128-bit key stream. Key generation algorithm generate two round keys for every round and this sub key added to input selected from buffers in add round step in non-linear function.

NLFS have two modes basic mode that is synchronous mode and self synchronous mode. In synchronous mode key stream is independent of plain text and cipher text, in self-synchronous mode key stream generation depending on cipher text. In self synchronous mode generated key-stream update first

512-bit buffer-A and cipher text update the second buffer-B, remain everything is same in self-synchronous mode.

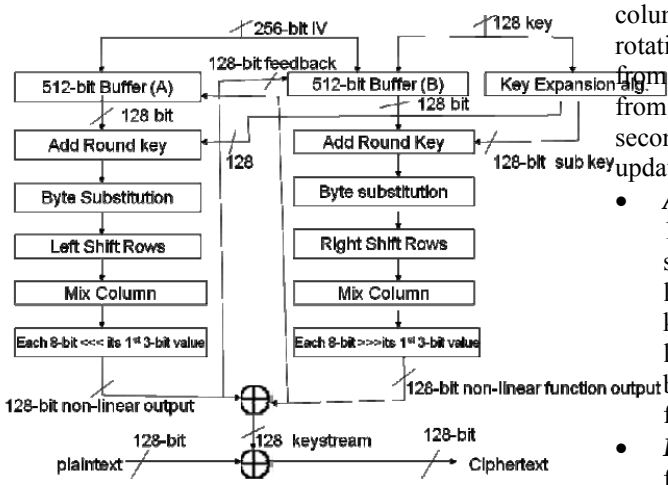
Symbol	denotes
⊕	XOR
<<<	Bit-wise left rotation
>>>	Bit-wise right rotation
Prefix '0x'	Hexa decimal numbers.

Symbol table

In this stream cipher each unit size is 8-bit and buffer size is 512-bit so it has 64 units. And total internal state has 128 units.

In encryption side and decryption side key stream generator is same and in same order key-stream sequence generates. In encryption side each 128-bit plaintext XOR with 128- bit generated key-stream, generates 128-bit cipher text. in decryption side each 128-bit cipher text XOR with 128-bit cipher text XOR with 128-bit key-stream and generates 128-bit plain-text.

- ***Input parameters:*** NLFS takes two inputs 256-bit initial vector and 128-bit key from user. Also user has to select two s-boxes from available 16 primitive polynomial generated S-boxes. A new initial vector (IV) is used for each new message.
- ***Buffer initialization:*** This stream cipher uses huge internal state 1024-bits and these 1024 –bits divided into two buffers A and B. Buffer A initialized by 256-bit initial vector. Buffer B initialized by both initial vector and 128-bit key.



Synchronous mode

- ***Buffers updation:*** Internal state updation mainly affect the security of stream cipher. NLFS efficiently update internal state by two non-linear functions output. Buffer A updated by 2nd non-linear function output and buffer B updated by 1st non-linear function output. Internal state of NLFS is updated by non-linear function output nearly all bits affected and full updation of internal state prevents guess and determine (GD) attack.
- ***Key-stream generation*:** key-stream is get from XOR of two non-linear functions 128-bit output.
- **Key expansion algorithm:** 128-bit key initializes key expansion algorithm. This is similar to AES key expansion but here every round takes 2 different sub-keys gives to two non-linear functions.
- ***Initial setup rounds*:** Before generating first 128-bit key-stream, NLFS key-stream generator needs to perform 16 initial rounds and here internal state updated by non-linear functions output and Key Expansion algorithm generate 3 sub-keys for every round. These 16 initial setup rounds avoid key and initial vector leakage.
- **Cipher text generation:** 128-bit plaintext is taken from message and 128-bit key-stream is taken from key-stream generation. NLFS output function performs XOR of those two and generates 128-bit cipher text. If message is not multiple of 128-bit then padding will takes place i.e. last block not have 128-bits and remaining bits filled by blank spaces.

4.1 Non-linear function

Non-linear function of NLFS have AES non-linear function steps add-round key, byte substitution, mix column, shift rows and it extra includes value-based rotation. First non-linear function takes filtered 128-bit from first buffer-A and second non-linear functions from second buffer-B. first ,non-linear function update second buffer-B and second non-linear function update first buffer-A.

- ***Add round key step*:** it computes XOR of filtered 128-bit of 512-bit buffer and 128-bit sub key supplied by key expansion algorithm. Two non-linear functions take two separate 128-bit sub keys from key expansion algorithm. First non-linear function take filtered 128-bit from first buffer-A and second non-linear function take filtered 128-bit from second buffer-B.
- ***Byte substitution*:** in this step two non-linear functions use two different primitive polynomial generated S-boxes, instead of AES irreducible polynomial (x8+x4+x3+x+1) generated S-box. Primitive polynomial generated s-box is secure than AES s-box and 16 primitive polynomials existing with degree 8 so 16 primitive polynomial generated boxes are available. Selection of s-boxes automatically done by 128-bit secret key is last 4-bit nibble.

- ***Shift rows*:** it takes 4X4 byte matrix from byte substitution step.

1. In first non-linear function, 1st row no change, 2nd row one circular left shift, 3rd row two circular left shift and 4th row three circular left shift.
2. In second non-linear function 1st no change, 2nd row one circular right shift, 3rd row two circular right shifts and 4th row 3 circular right shift.

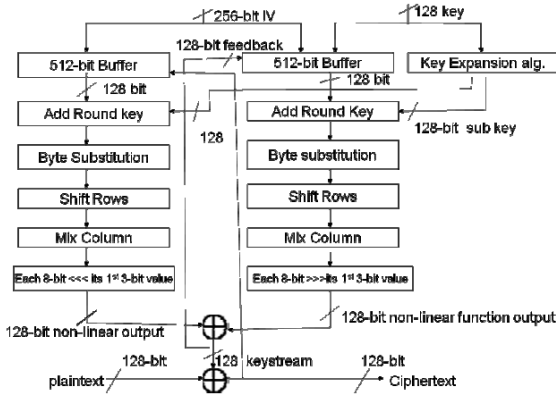
- ***Mix column:*** This step operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column. Same MDS (maximum distance separable) is used both non-linear functions. In this step performs multiplication of state matrix and MDS matrix.
- ***Value based rotation step*:** in this step it shuffles bits with in byte and shift rows step shuffle bytes. It takes each 8-bit unit and it is rotate by its first 3-bit (decimal) value. Value based rotation step first and second non-linear functions left and right rotates each 8-bit respectively. For example "10100110" word left rotate by 5(101) times in first nonlinear function and in second non-linear function "11001110" right rotated by 6(110).

4.2 Self synchronous mode of NLFS

NLFS have two modes.

1. **Synchronous mode**: NLFS's key-stream generation is independent of plaintext and cipher text.
2. **Self-synchronous mode**: NLFS's key-stream generation is depending on cipher text.

In self synchronous mode first buffer-A is updated by cipher text and second buffer is updated by key-stream in encryption side. Where as decryption side, buffer-A is updated by plaintext and second buffer is updated by key-stream. In this mode internal state is more efficiently update than synchronous mode.



Self synchronous mode.

This self-synchronous mode is useful and efficient than synchronous mode in error prone channels. Self synchronous mode face problem with error propagation i.e. if during transmission any cipher text bit changes that will not give correct plaintext and this incorrect plaintext uses by key-stream generator decryption side. Then key-stream generator generates wrong pseudo key-streams and we don't get correct plaintext. Whenever receiver correctly receives 128-bit cipher text, then only receiver able to generate next key-stream correctly.

4.3 Primitive polynomials generated S-boxes

- ***Irreducible polynomial:*** A polynomial is said to be irreducible if it cannot be factored into nontrivial polynomials over the same field.
- ***Primitive polynomial:*** is a polynomial that generates all elements of an extension field from a base field.

Primitive polynomial is the minimal polynomial of a primitive element of the extension field GF(p^m). A primitive polynomial must have a non-zero constant term, for otherwise it will be divisible by *x*. Over the field of two elements, all primitive polynomials have an odd number of terms, otherwise they are divisible by *x+1*. All primitive polynomials are irreducible polynomials.

Total number of primitive polynomials in GF (p^m) with degree m = $\varphi(\mathbf{p^m - 1})/\boldsymbol{m}$

NLFS uses primitive polynomial generated S-box and AES uses irreducible

polynomial ($x^8+x^4+x^3$ +x+1=0) generated S-box. There exist 16 primitive polynomials

with degree 8 under GF (2). These are

1) $x^8+x^4+x^3+x^2+1$
2) $x^8+x^6+x^5+x^3+1$
3) $x^8+x^7+x^6+x^5+x^2+x+1$
4) $x^8+x^5+x^3+x+1$
5) $x^8+x^6+x^5+x^2+1$
6) $x^8+x^6+x^5+x+1$
7) $x^8+x^7+x^3+x^2+1$
8) $x^8+x^5+x^3+x^2+1$
9) $x^8+x^6+x^4+x^3+x^2+x+1$
10) $x^8+x^7+x^6+x+1$
11) $x^8+x^7+x^5+x^3+1$
12) $x^8+x^7+x^2+x+1$
13) $x^8+x^6+x^3+x^2+1$
14) $x^8+x^7+x^6+x^3+x^2+x+1$
15) $x^8+x^7+x^6+x^5+x^4+x^2+1$.
16) $x^8+x^6+x^5+x^4+1$.

In NLFS, these 16 primitive polynomials generated S-boxes available and user has to select two S-boxes for two non-linear functions.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	99	30	1E	AD	CA	B5	DD	77	04	78	B7	CD	08	7F
1	49	31	1C	9F	D0	40	EE	5F	09	BC	41	6F	A3	6A	18	12
2	03	B3	4A	E5	DC	B0	68	9B	BA	1B	87	8F	A5	67	7D	E0
3	23	9E	8C	0B	07	37	10	C5	76	88	E7	D2	AB	F3	AE	DB
4	26	F4	7E	32	82	73	20	13	C9	47	FF	3A	E6	0C	1F	B2
5	FA	BE	2A	91	64	1D	60	D9	00	01	14	4F	19	34	D7	F6
6	43	D6	E8	8A	E1	B8	22	4D	51	6D	3C	6E	AF	FC	45	8B
7	9C	44	96	69	21	AA	BB	39	72	BF	5E	4B	85	61	3F	F9
8	B4	EA	A8	F8	98	59	CB	C4	93	50	6B	8E	C2	84	2E	92
9	36	7B	71	81	2D	9A	CF	F1	A1	55	D4	B1	5D	2F	FE	24
A	DA	33	8D	56	C7	9D	1A	3D	95	79	5C	58	E2	5A	3E	D5
B	A7	3B	27	A2	D8	6C	75	16	2B	D1	BD	C0	4C	C1	A9	83
C	06	CE	CC	48	D3	0A	97	89	57	15	FB	0E	C3	E3	74	35
D	0F	2C	11	ED	B9	A6	90	46	05	52	AC	80	70	54	17	C6
E	E9	EF	F0	C8	EC	F5	66	F7	42	86	F2	E4	7A	DE	4E	A0
F	EB	A4	0D	28	FD	94	02	29	65	53	62	25	38	DF	5B	B6

$x^8+x^6+x^5+x^3+1$ polynomial generated S-box.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	18	31	2A	0A	4A	FA	C7	EB	23	AD	03	3A	5B	BB
1	C5	D4	D3	E8	43	50	04	54	A7	1D	CF	1B	8B	7E	FB	F9
2	C4	EA	4C	85	3B	86	52	97	87	FD	0E	B6	D0	4B	F8	1C
3	01	F2	5C	F1	C1	1A	AB	6B	17	D6	19	14	DB	9F	DA	8D
4	44	C6	53	A1	F4	9B	E4	37	4F	EE	65	57	0F	4E	ED	71
5	E5	21	2C	B4	D5	B5	89	C9	BA	3C	83	69	5A	CD	DC	EC
6	A6	26	5F	BC	FC	99	DE	12	32	68	2B	60	F3	EF	67	98
7	59	11	4D	DD	AA	29	D8	84	3F	10	E9	B1	BF	77	E0	82
8	F0	C3	45	CE	8F	90	F6	6F	A8	06	1F	15	A0	2D	BD	AF
9	75	B8	A5	B2	94	09	79	A3	55	3E	F5	80	24	E3	6A	02
A	20	51	42	07	30	8E	88	C2	CC	B3	08	96	16	61	36	CA
B	7B	6D	38	22	13	FF	66	40	0B	B7	C0	3D	48	62	A4	D9
C	81	7F	35	00	7D	7A	8C	9C	AC	F7	1E	6E	49	A2	2F	6C
D	CB	92	E6	28	47	39	E2	78	DF	27	25	E7	95	D7	9E	34
E	8A	41	AE	70	74	33	C8	5E	73	91	46	A9	BE	9A	64	E1
F	B9	58	2E	5D	D2	D1	FE	72	0D	05	9D	0C	56	B0	93	76

$x^8+x^7+x^6+x^5+x^2+x+1$ polynomial generated S-box

5. RANDOMNESS TESTING OF THE NLFS PSEUDO-RANDOM KEY STREAM GENERATOR (PRKG)

In this section we checking the randomness of NLFS key stream generator. Generally any PRKGs must satisfy following two requirements:

1. The output sequence has good enough randomness.
2. Any two output sequences generated by different initial data are significantly different.

5.1 Repetition test

In this method we check randomness of 128-bit key-stream. I had taken 40,000 key streams (2^{22} bits) as sample. Chosen key-stream one by one and compare it with all other key-streams. None of the key-stream repeated.

5.2 Frequency testing of PRKG

In this section we present the results of our statistical randomness tests. The tests we used to examine the randomness of the output sequences of NLFS. The *frequency test* checks that there is an equal proportion of ones and zeros in the bit stream. For randomness the proportion of ones and zeros in the bit stream should be approximately equal, since any substantial deviation from equality could result in a successful cryptanalytic attack on the cipher. For example, assume that a cryptanalyst attacking the stream cipher knows the type of plaintext being used, e.g. standard English text coded in 8-bit ASCII, and the keystream has 43 of the bits zero. Under this assumption the cryptanalyst knows the frequency distribution of the plaintext in terms of single bits, digraphs and trigraphs. With this knowledge the cryptanalyst could recover a substantial amount of the plaintext, using ciphertext alone.

- We are taking 40,000 key-streams as sample.

5.2.1 1-bit frequency test

In this method, we check '0' and '1' bits occurrences in above sample. If either '0' or '1' bit fully dominate in generated key stream then NLFS pseudorandom generator is not good. '0' bit occurrences 47.2 % and '1' bit occurrences 52.8 %.

- NLFS pseudorandom generator satisfies the 1-bit frequency test.

5.2.2 2-bit frequency test

In this method, we count number of occurrences of 00, 01, 10 and 11 in large amount
of key stream. If any bit pair fully dominate in generated keystream then NLFS
pseudorandom generator is not good.

2-bit	Occurrence %
00	28.3
01	26.4
10	23.6
11	22.7

2-bit frequency test table.

- NLFS pseudorandom generator satisfies the 2-bit frequency test.

5.2.3 4-bit frequency test

In this method, we count number of occurrences of 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 and 1111 in large amount of key stream.

4-bit	**Occurrence %**
0000	**6.32**
0001	**6.57**
0010	**5.96**
0011	**5.63**
0100	**6.21**
0101	**5.73**
0110	**5.69**
0111	**6.45**
1000	**6.68**
1001	**5.93**
1010	**6.29**
1011	**6.82**
1100	**5.87**
1101	**6.34**
1110	**6.83**
1111	**6.68**

4-bit frequency test table.

- NLFS pseudorandom generator satisfies the 4-bit frequency test.

5.2.4 8-bit frequency test

This method also similar to above frequency testing methods, here also we count all combinations of 8-bits. Check whether all are equally distributed or not.

- NLFS pseudorandom generator satisfies the 8-bit frequency test.

5.3 Testing the efficiency of the PRKG

Here we checks PRKG second necessary condition i.e. any two output sequences generated by different initial data are significantly different.

5.3.1 Test the impact of initial vector on PRKG

In this method, we test the 256-bit initial vector impact on pseudorandom key stream generator by fixing the key and varying initial vector. 1500 different initial vectors applied on pseudorandom key stream generator and every time it generates different key stream sequence. PRKG efficiently using the initial vector.

5.3.2 Test the impact of key on the PRKG.

In this method, we test the key impact on pseudorandom key stream generator by fixing the initial vector and varying key. 1500 different keys applied on pseudorandom key stream generator and every time it generates different output (key stream) sequence. NLFS key stream generator efficiently uses the key.

6. Security of NLFS stream cipher

Generally stream ciphers are weaker than block ciphers because they focus on fastness and so they reduce complex implementation for fastness. This section covers how NLFS secure from Shannon theory, linear cryptanalysis and differential cryptanalysis.

6.1 Security proved by Shannon theory

From above randomness testing methods proved NLFS's pseudorandom key stream generator efficiently (very high randomly) generates key stream

Shannon theory: a cipher has perfect secrecy if $p[x/y]=p[x]$ for all $x \in$ plaintext, for all $y \in$ ciphertext.

NLFS is PRKG acts as nearly true random generator, it generates random key stream. Cipher text is generated from plaintext and random key stream by XOR operation. So NLFS generated ciphertext is random i.e. independent of plaintext. So cryptanalyst never gets plaintext from ciphertext, even if entire ciphertext is available. From above Shannon theory we can say NLFS is nearly perfect secrecy.

6.2 Linear Cryptanalysis

Linear cryptanalysis is a method to reconstruct secret key by solving linear equations of key stream generators of stream ciphers. Linear cryptanalysis studies the correlation between linear combinations of input and output bits of functions. Linear cryptanalysis is known-plaintext attack.

NLFS stream cipher's two non-linear functions, two efficiently (dynamically) updating buffers and value based rotation step, eliminate linear correlation between input plaintext) and output (ciphertext). Complex non-linear function of NLFS prevents linear relationship between plaintext and ciphertext.

In this stream cipher we are using 16 primitive polynomial generated S-boxes and all these S-boxes listed out in section 2.3. S-box is non-linearity can express by linear approximation equations between its

input and output. As like AES S-box, NLFS S-box also taking 8-bit input X and gives 8-bit output Y. we can form linear equations between input bits(x1, x2, x3, x4, x5, x6, x7 and x8) and output bits(y1,y2,y3,y4,y5, y6, y7 and y8). In linear cryptanalysis we have to find linear approximation equation with higher probability and bias. All linear equations between 8-bit input and 8-bit output can show linear approximation table, but linear approximation table size is 256X256 for 8X8 S-box. We can't show all linear equations biases in table. Success of cryptanalysis depending on larger magnitude in linear approximation table.

Linear equation	Bias	probability
$x_8 = y_1 \oplus y_5$	+14 / 256	142 / 256
$x_8 = y_1 \oplus y_4$	+14 / 256	142 / 256
$x_7 = y_2 \oplus y_3 \oplus y_4 \oplus y_5$	+14 / 256	142 / 256
$x_7 = y_1 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_8$	+14 / 256	142 / 256
$x_7 \oplus x_8 = y_1 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_8$	+14 / 256	142 / 256
$x_7 \oplus x_8 = y_2 \oplus y_5 \oplus y_8$	+14 / 256	142 / 256
$x_6 \oplus x_8 = y_2 \oplus y_3 \oplus y_4 \oplus y_5$	+14 / 256	142 / 256
$x_6 \oplus x_7 = y_2 \oplus y_3 \oplus y4 \oplus y_5$	+14 / 256	142 / 256
$x5 \oplus x_8 = y_2 \oplus y_3 \oplus y_5 \oplus y_6 \oplus y_8$	+14 / 256	142 / 256
$x2 \oplus x_6 = y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7$	+14 / 256	142 / 256
$x_8 = y_5 \oplus y_6$	+12 / 256	140 / 256
$x_7 = y_5 \oplus y_7 \oplus y_8$	+12 / 256	140 / 256
$x4 \oplus x_6 \oplus x_8 = y_5 \oplus y_6 \oplus y_7$	+12 / 256	140 / 256
$x_3 \oplus x_5 \oplus x_7 = y_2 \oplus y_4 \oplus y_7$	+12 / 256	140 / 256
$x_1 \oplus x3 \oplus x_5 = y_2 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_8$	+12 / 256	140 / 256
$x_5 \oplus x_6 \oplus x_7 \oplus x_8 = y_2 \oplus y_6$	-14 /256	114 / 256
$x_4 = y_1 \oplus y_2 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_8$	-14 /256	114 / 256
$x_4 \oplus x_5 \oplus x_6 \oplus x_7 = y_1 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_8$	-14 /256	114 / 256
$x_3 \oplus x_4 \oplus x_7 = y_1 \oplus y_6 \oplus y_7$	-14 /256	114 / 256
$x_3 \oplus x_5 \oplus x_7 = y_1 \oplus y_2 \oplus y_7$	-14 /256	114 / 256

Linear approximation table for 2nd S-box

7. CONCLUSION

Period of stream cipher depending on number of possible states of internal state during key-stream generation and finding exact period of NLFS is very difficult.

If stream cipher internal state contains independent linear feedback shift egisters
(LFSR), then we can calculate period by using formula
Period=2^{32}x -1.

Here assumption LFSR size is 32-bit and 'x' is number of LFSRs in internal state. NLFS internal state initialized by 128-bit key and 256-bit initial vector. After initialization, internal state has one of possible O (2^{384}) initial states.
Lower bound of period= $\Omega(2^{384})$.

In best case, internal state may have all possible 1024-bit states.
Upper bound of period= O (2^{1024}).

We believe large and efficiently updating internal state resist Guess and Determine attack on NLFS and two non-linear functions output fully updating the internal state. That efficient updating of internal state resists GD attack on NLFS. NLFS have enough period to prevent distinguish attack.

Our goal is, design a stream cipher which is fast and provide security as much as AES
block cipher, to meet this objective we use AES is non-linear function and key expansion algorithm in NLFS. In AES 10 rounds are there, but in NLFS has one round, in which two non-linear functions used. So NLFS can't provide that much of AES security. In AES key expansion algorithm generates 11 sub-keys and NLFS key expansion algorithm generates 3 sub-keys for one plaintext block encryption.

8. REFERENCES

1) "*A tutorial on linear and differential cryptanalysis*" by Howard M.Heys

2) "*Two attacks against the HBB stream cipher*" Antoine joux and Frederic muller

3) "*Cryptography and network security*" by William Stallings.

4) "*Evaluation of the MUGI Pseudo-Random Number Generator*" by Ed Dawson, Gary Carter, Helen Gustafson and Matt Henricksen.

5) "*Crypt Stream cipher*" by Makoto Matsumoto, Mutsuo Saito, Takuji Nishimura and Mariko Hagita.

6) "*International standards for stream ciphers*" by Chris J. Mitchell and Alexander W. Dent.

7) "*Turing stream cipher*" by Greg Rose, Philip Hawkes

8) "*Correlation Attacks on Stream Ciphers*" by W T Penzhorn
.

9) http://www.theory.csc.uvic.ca/~cos/inf/neck/PolyInfo.html

10) "*Ongoing Research Areas in Symmetric Cryptography*" by Daniel Augot.

11) "*A tutorial on linear and differential cryptanalysis*" by Howard M.heys.

12) "*Relating differential distribution tables to other properties of substitution boxes*" by Xian-Mo Zhang.

13) "*On linear approximation tables and ciphers secure against linear cryptanalysis*" by Luke Conner.

14) "*Open problems related to algebraic attacks on stream ciphers*" by Anne Canteaut.

15) "*Analysis of the non-linear part of Mugi* " by Biryukov and adi shamir.

16) *http://www.ecrypt.eu.org/*