SYNOPSIS

NLFS: A New Non-Linear Feedback Stream Cipher

Submitted by
Siva Apparao Rapeti
Roll No: 06CS6021
Computer Science and Engineering

Under Guidance of
Prof: D. Roy Chowdhury
Computer Science and Engineering

Department of Computer Science & Engineering,
Indian Institute of Technology,
Kharagpur-721302,
West Bengal, India.
Abstract

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 $S$-boxes 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. In value based rotation it rotates each 8-bit word by its first 3-bit (decimal) value.

NLFS takes 256-bit Initial vector and 128-bit key. NLFS contain two 512-bit buffers both are non-linear buffers i.e. both are updated by non-linear functions output. First buffer initialized by 256-bit initial vector and second 512-bit buffer initialized by initial vector and key. The outputs of non-linear functions are divided into 8-bit words and they rotate by their first 3-bit (decimal) value in value-based rotation step in non-linear function. The outputs of both non-linear functions are XORed 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 and 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.
Contents
1. Introduction to NLFS.
   1.1 Motivation
   1.2 Contribution
2. NLFS stream cipher structure
   2.1 Non-linear function
   2.2 primitive polynomial generated S-box
3. Self-synchronous mode of NLFS.
4. Randomness testing of NLFS key-stream generator.
   4.1 Frequency test
   4.2 Testing the efficiency of the PRKG.
      4.2.1 Test the impact of initial vector on PRKG.
      4.2.2 Test the impact of key on the PRKG.
5. Security of NLFS
   5.1 NLFS security proved by Shannon theory
   5.2 Cryptanalysis of NLFS
      5.2.1 Linear cryptanalysis
      5.2.2 Differential cryptanalysis.
6. Conclusion.
7. References.
1. Introduction to NLFS.

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. A stream cipher generates key stream (a sequence of bits used as a key). 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 ciphertext, 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.

Stream ciphers main requirements are fastness (i.e. high throughput), 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 (performance), flexibility and it needs to meet outside world requirements.

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 is broken 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 determine attack, which has 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.

The ECRYPT NOE(European Network of Excellence for Cryptology) collect the software application stream ciphers in profile 1 and 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.

1.1 Motivation

Above mentioned stream ciphers (SCREAM, PY, SNOW, SOSEMANUK and GRAIN) are fast but broken. Above ciphers use non-linear functions in their implementation, but those non-linear 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.

1.2 Contribution

By using AES non-linear function and key generation, I have proposed the design of new stream cipher called Non-linear feedback stream cipher (NLFS). NLFS takes two similar AES non-linear functions and these two proceed parallelly to generate key-
2. Design of Non-Linear Feedback stream cipher (NLFS)

NLFS (non-linear feedback stream cipher) is a fast and secure stream cipher for Software applications.

**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 IV is used for each new message.

**Buffer initialization:** NLFS have two 512-bit buffers A and B (refer to figure 1). Buffer A initialized by initial vector. Buffer B initialized by both initial vector and key.

**Buffers updation:** Internal state updation mainly effect the security of stream cipher. So 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. Present state and next state of internal state are fully different i.e. completely updated for every round.

NLFS synchronous mode

Figure 1
**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 for two non-linear functions.

**Initial setup rounds:** Before generating first key-stream, NLFS key-stream generator needs to perform 4 initial rounds. These 4 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.

### 2.1 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.

**Add round key step:** it computes XOR of filtered 128-bit of 512-bit buffer and sub key supplied by key expansion algorithm. Two non-linear functions take two separate sub keys from key expansion algorithm.

**Byte substitution:** in this step two non-linear functions use two different primitive polynomial generated S-boxes, instead of AES irreducible polynomial generated S-box.

**Shift rows:** it takes $4 \times 4$ byte matrix from byte substitution step.
- 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.
- In 2nd non-linear function 1st no change, 2nd row one circular right shift, 3rd row two circular right shift 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:** 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.

### 2.2 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 $\text{GF}(p^m)$. In other words, a polynomial $F(X)$ with coefficients in $\text{GF}(p) = \mathbb{Z}/p\mathbb{Z}$ is a primitive polynomial if it has a root $\alpha$ in $\text{GF}(p^m)$ such that $\{0, 1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{p^{m-2}}\}$ is the entire field $\text{GF}(p^m)$, and moreover, $F(X)$ is the smallest degree polynomial having $\alpha$ as root. In ring theory, the term *primitive polynomial* is used.
for a different purpose, to mean a polynomial over a unique factorization domain (such as the integers) whose greatest common divisor of its coefficients is a unit.

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 \( x+1 \).

Total number of primitive polynomials in \( \text{GF}(p^m) \) with degree \( m \),
\[
\varphi(p^m - 1)/m
\]
All primitive polynomials are irreducible polynomials.

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

\[
\begin{align*}
1) & \quad x^8+x^4+x^3+x^2+1 \\
2) & \quad x^8+x^6+x^5+x^3+1 \\
3) & \quad x^8+x^7+x^6+x^5+x^2+x+1 \\
4) & \quad x^8+x^5+x^3+x+1 \\
5) & \quad x^8+x^6+x^5+x^2+1 \\
6) & \quad x^8+x^6+x^5+x+1 \\
7) & \quad x^8+x^7+x^3+x^2+1 \\
8) & \quad x^8+x^7+x^3+x^2+1 \\
9) & \quad x^8+x^6+x^4+x^3+x^2+x+1 \\
10) & \quad x^8+x^7+x^6+x+1 \\
11) & \quad x^8+x^7+x^5+x^3+1 \\
12) & \quad x^8+x^7+x^5+x^3+x+1 \\
13) & \quad x^8+x^6+x^3+x^2+1 \\
14) & \quad x^8+x^7+x^6+x^3+x^2+x+1 \\
15) & \quad x^8+x^7+x^6+x^5+x^4+x^2+1 \\
16) & \quad x^8+x^6+x^5+x^4+1 .
\end{align*}
\]

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

3. Self synchronous mode of NLFS.

NLFS have two modes synchronous mode and self synchronous mode.

- **synchronous mode**: NLFS’s key-stream generation is independent of plaintext and ciphertext.
- **self-synchronous mode**: NLFS’s key-stream generation is depending on ciphertext.

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.

This self-synchronous mode is useful and efficient than synchronous mode in error prone channels. Self synchronous mode suffer from error propagation i.e. if during the 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 ciphertext, then only receiver able to generate next key-stream correctly.
4. Randomness testing of the NLFS Pseudo Random Keystream Generator (PRKG).

In this section we checking 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.

4.1 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.

4.1.1 1-bit frequency test.

In this method, we check ‘0’ and ‘1’ bits occurrence in large amount of key stream. If either ‘0’ or ‘1’ bit fully dominate in generated key stream then NLFS pseudorandom generator is not good.

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

4.1.2 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.
• NLFS pseudorandom generator satisfies the 2-bit frequency test.

4.1.3 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. If any one fully dominate in generated keystream then NLFS pseudorandom generator is not good.
• NLFS pseudorandom generator satisfies the 4-bit frequency test.

4.1.4 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.

4.2 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.

4.2.1 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 key and varying initial vector.
Large number of different initial vectors applied on pseudorandom key stream generator and every time it generates different output (key stream).
4.2.2 Test the impact of key on the PRKG.
In this method, we test the key impact on pseudorandom key stream generator by fixing initial vector and varying key.
Large number of different keys applied on pseudorandom key stream generator and check whether every time it generating the different output (key stream) sequence or not.

5. 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.

5.1 Security proved by Shannon theory
From above randomness testing methods proved NLFS’s pseudorandom key stream generator efficiently (very high randomly) generates keystream.

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 keystream. 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.
5.2.1 Linear cryptanalysis.

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).

In primitive S-box linear approximation table, best magnitude is 14 and this is not enough for linear cryptanalysis. In linear approximation table at least 16 magnitude is needed from [11], here this magnitude specifies bias for cryptanalysis. And sixteen initial setup rounds are there before generating key-stream then the above bias goes too low. Linear cryptanalysis is not possible on NLFS.

5.2.2 Differential cryptanalysis.

In differential cryptanalysis, an input difference produces an output difference with a certain probability, which if large enough can often lead to the determination of key. Differential cryptanalysis is a chosen plaintext attack.

NLFS uses two non-linear functions so totally 32-times S-box substitute for a single round i.e. for generation of 128-bit key stream. NLFS byte substitution step provide very high confusion. Similarly MDS (maximum distance separable) is used 32-times for a single round in Mixcolumn step in NLFS. This Mixcolumn step provides very high diffusion.

Generally Stream ciphers security mainly depending on updation of internal state. NLFS contain large internal state 1024-bit and it is efficiently and dynamically updated by two highly complex non-linear functions output.

In S-box difference distribution table, largest value is 4 and differential cryptanalysis bias value for single S-box is 4/256. This bias not enough for differential cryptanalysis on NLFS.

6. Conclusion

NLFS satisfies the all randomness tests and it is easy to implementation. NLFS construction based on two similar AES non-linear functions, AES key generator and 1024 bit internal state. It prevents linear and differential attacks, we have to check whether it is prevent remain attacks (algebraic, guess and determine, distinguish attacks …) or not.
7. References
1) http://www.ecrypt.eu.org/
2) “A tutorial on linear and differential cryptanalysis” by Howard M.Heys
3) “Two attacks against the HBB stream cipher” Antoine joux and Frederic muller
6) “Cryptmt Stream cipher” by Makoto Matsumoto, Mutsuo Saito, Takuji Nishimura and Mariko Hagita.
7) ”International standards for stream ciphers” by Chris J. Mitchell and Alexander W. Dent.
8) “Turing stream cipher” by Greg Rose, Philip Hawkes
9) “Correlation Attacks on Stream Ciphers” by W T Penzhorn.
10) http://www.theory.csc.uvic.ca/~cos/inf/neck/PolyInfo.html
11) “Ongoing Research Areas in Symmetric Cryptography” by Daniel Augot