An Improved Stream Cipher Algorithm Based on NFLSR

Guanghui Feng, Chunfu Zhang*
Zhuhai College of Jilin University, Zhuhai, 519041, China
14139339@qq.com

Abstract. To overcome defects of traditional stream cryptographic algorithms based on XOR operation, this paper proposes a new stream cipher algorithm with higher security and suitability for encrypting short information, which is based upon traversal operation and zipper-like random dynamic hash mapping, has the same time and space complexity as the traditional one. It generates the key stream through an application of Nonlinear Feedback Shift Register (NLFSR) and Bits Scrambling.

1. Introduction
The current study on stream cipher mainly concentrates on key stream generator and methods of attacking the key stream. In fact, the key stream generator would form the key stream involving 0, 1 data streams through a given algorithm, which is usually based on mathematical models that can generate pseudo-random sequences, such as Linear Feedback Shift Register (LFSR) [1][2], Clock Control Sequence[3], Chaos Theory [4][5], etc. The sender encrypts the plaintext with the key stream to generate ciphertext and sends it to the receiver, who decrypts the ciphertext with the same key stream to regain the plaintext. After years of research, many scholars have mastered many identifying and decoding methods. For instance, Courtois and Meier[6] in 2003 applied algebraic attacks to stream cipher algorithms on the basis of LFSR, greatly impacting on traditional stream cipher systems.

In order to resist related and algebraic attacks, the stream cipher algorithm based on LFSR needs more complex feedforward function. NLFSR[7][8], as a pseudo-random sequence generator with high security, can effectively resist correlation and algebraic attacks. Since 2004, key stream generators based on NLFSR technology have started to emerge. For example, the Grain, Trivium and MICKEY algorithms finally recommended by eSTREAM Project all use NLFSR as a key stream generator.

2. Problems
The stream cipher is used to transform plaintext into ciphertext in binary bits according to certain rules. With common stream cipher technology, ciphertext can be obtained through XOR operation between key stream and plaintext; the plaintext can be decrypted after regaining the original binary characters by XOR operation between the key stream and the plaintext.

The key of common stream cipher technology is to generate (pseudo) random sequences as key streams through various key stream generators. Therefore, attacks on stream cipher also focus on attacks on key stream generators.

One of the defects of this approach is the single encryption and decryption process. Now there have been many attack technologies on the key stream generator in that the security of the stream cryptosystem depends entirely on the security intensity of the key stream. Another defect is that ciphertext with good randomness and code distribution uniformity cannot be obtained just by XOR operation.
3. Method

Similar to LFSR, NLFSR is also based on gate circuits. From the perspective of algebraic expression, XOR gate is expressed as binary addition (+), while AND gate is expressed as binary multiplication (×). The difference between NLFSR and LFSR is that the feedback logic of NLFSR is composed of XOR gate and AND gate, while that of LFSR is composed of XOR gate only, which leads to higher complexity of NLFSR. Up to now, there is no universally applicable algebraic attack method against NLFSR. The output sequence of NLFSR is periodic, for seed keys with a length of n, the longest cycle of the output sequence can reach $2^n-1$. Obviously, the longer the cycle is, the better the encryption is.

We have figured out an encryption and decryption method based on zipper-like dynamic hash and NLFSR. The key stream generator is formed based on NLFSR and Boolean function with Bits Scrambling. The seed key is pre-processed by the key stream generator and then generates the output sequence as the key stream. In the process of traversing operation, the pseudo-plaintext is obtained from plaintext based on scrambling bits of key stream. At the same time, the zipper-like dynamic directly-addressing method based on key stream can be used to calculate the hash address of binary bits of pseudo-plaintext in ciphertext space, and then the pseudo-plaintext is mapped into ciphertext space to generate the final ciphertext. An illustrative diagram of a modified stream cipher system based on NFLSR is shown in Figure 1.

![An illustrative diagram of a stream cipher system based on a modified NFLSR](image1)

Fig.1 An illustrative diagram of a stream cipher system based on a modified NFLSR

![Nonlinear Feedback Register F1 (x₀, x₃, x₆, x₉, x₁₃)](image2)

Fig.2 Nonlinear Feedback Register F₁ \((x₀, x₃, x₆, x₉, x₁₃)\)

The encryption algorithm is expressed by natural language mixed with C ++ language. Its execution process is as follows:
(1) Read the file of plaintext and set the seed key, establish working pointers M, K and C, allocate storage space for plaintext, key and ciphertext, wherein M is the working pointer of plaintext, K is the working pointer of key, C is the working pointer of ciphertext, and count the M bytes (byte number of plaintext M).

(2) Calculate the output sequence of seed key by nonlinear feedback function.

The function void get_NLFSR_sequence (string str, int a, int b, int c, int d) calculates the first output sequence of the seed key by the nonlinear feedback function and stores it in the string - NlfsrSequence. Parameter str is the seed key, and a, b, c and d are the coefficients of the nonlinear feedback function $F_1(x_0, x_a, x_b, x_c, x_d) = x_0 \times x_a \times x_b \times x_c \times x_d$, as shown in the figure 2, where "⊕" is "XOR" operation (i.e. Modulo 2 addition) and "•" is "AND" operation (i.e. Modulo 2 multiplication). The algorithm is described as follows:

```cpp
bitset<N> SeedKey(str);
bitset<N> TempKey(SeedKey);
string TempSequence, NlfsrSequence;
do{
    j = SeedKey[0] ⊕ SeedKey[a] ⊕ SeedKey[b] ⊕ (SeedKey[c] * SeedKey[d]);
    SeedKey.operator>>=(1);
    SeedKey[N - 1] = j;
    TempSequence = SeedKey.to_string();
    NlfsrSequence.push_back(TempSequence[0]);
}while (TempKey.to_string() != SeedKey.to_string());
```

(3) Cyclic filling of the aforementioned output sequence NlfsrSequence to the key stream space K.

The function void get_key_stream (string Seq) converts the output sequence Seq by the nonlinear feedback function into a string, which is stored in the key space K by cyclic filling. The parameter Seq is the output sequence generated by the seed key by the nonlinear feedback function. The algorithm is described as follows:

```cpp
for(i=0; i<M_bytes;i++){
    x=0;
    for(j=0; j<8; j++) x=2*x+Seq[count%Seq.length()] - 48;
    K[i] = x;
}
```

(4) The pseudo plaintext stream $M'$ is obtained by scrambling bits of key stream and plaintext stream M, where $F_2()$ is a Boolean function with Bits Scrambling. The algorithm is described as follows:

```cpp
for(i=0; i<8*M_bytes;i++){
    K_bit=read_one_bit(K,i);
    if(Z1 == (C1 + X1*i)%Y1 || Z2 == (C2 + X2*i)%Y2){
        if(1 == K_bit){
            write_one_bit_0(K,i); K_bit=0;
        }else write_one_bit_1(K,i); K_bit=1;
    }
    if(0 == K_bit){
        if(1 == read_one_bit(M,i)) write_one_bit_0(M,i);
        else write_one_bit_1(M,i);
    }
}
```

(5) The pseudo-plaintext $M'$ and the key stream $K$ are traversed in one process, and the $M'$ is mapped into the ciphertext space to obtain the ciphertext C according to the zipper dynamic hash rule (based on the N way merge algorithm).
By Function void en_zipper_hash (int road1, int road1), this algorithm adopts a two-way merge algorithm. Parameters road1 and road1 are the initial values of the working pointers and the starting position of splitting plaintext. The algorithm is described as follows:

```c
q1=road1,q2=road2,road=road1;count1=0,count2=0;
do{
    while(0 == read_one_bit(K,q1)) {q1=(q1+1)%(8*M_bytes); count1++;}
    M_road1 = read_one_bit(M,q1); q1=(q1+1)%(8*M_bytes);count1++;
    if(1 == M_road1) write_one_bit_1(C,road++);
    else write_one_bit_0(C,road++);
    while(1 == read_one_bit(K,q2)) {q2=(q2-1+8*M_bytes)%8*M_bytes; count2++;}
    M_road2 = read_one_bit(M,q2); q2=(q2-1+8*M_bytes)%8*M_bytes;count1++;
    if(1 == M_road2) write_one_bit_1(C,road++);
    else write_one_bit_0(C,road++);
}while(count1<8*M_bytes && count2<8*M_bytes && road<8*M_bytes);
while(q1!=road1 && road<8*M_bytes){
    while(0 == read_one_bit(K,q1) && q1!=road1) q1=(q1+1)%(8*M_bytes);
    int M_road1 = read_one_bit(M,q1); q1=(q1+1)%(8*M_bytes);
    if(1 == M_road1) write_one_bit_1(C,road++);
    else write_one_bit_0(C,road++);
}
while(q2!=road2 && road<8*M_bytes){
    while(1 == read_one_bit(K,q2) && q2!=road2) q2=(q2-1+8*M_bytes)%8*M_bytes;
    int M_road2 = read_one_bit(M,q2); q2=(q2-1+8*M_bytes)%8*M_bytes;
    if(1 == M_road2) write_one_bit_1(C,road++);
    else write_one_bit_0(C,road++);
}

Description:
(1) void write_bit_0 (char * p_str, int n) implements the function of generating 0 towards the nth binary bit, and the parameter p_str is a pointer to a string.
(2) void write_bit_1 (char * p_str, int n) implements the function of generating 1 towards the nth binary bit, and the parameter p_str is a pointer to a string.
(3) int read_one_bit (char * p_str, int n) implements the function of reading the nth binary bit and returning, and the parameter p_str is a pointer to a string.

4. Application example
Encrypted example:
Seed key:{1000}_2
NFLSR function: F1(x0,x1,x2,x3)=x0 ⊕ x1 ⊕ x2 ⊕ x2•x3
Plaintext: M="aaaa"={01100001,01100001,01100001,01100001}_2.
Key: K="4╗÷Ω"={00110100,10111011,11110110,11101010}_2.
Ciphertext: C="ö╨█♀"={10010100,11010000,11011011,00001100}_2.
Figure 3 shows the illustrative diagram of the above algorithm when the seed key is Binary \{1000\}2. The nonlinear feedback function is F1(x0, x1, x2, x3); the plaintext is "aaaa", and finally the ciphertext C is obtained.

\[
M=\text{"aaaa"} = \{0X61, 0X61, 0X61, 0X61\}
\]

\[
\begin{array}{ccccccccccccccccccccccccccccccccc}
6 & 1 & 6 & 1 & 6 & 1 & 6 & 1 \\
\hline
m_6 & m_1 & m_0 & m_6 & m_1 & m_0 & m_6 & m_1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

\[
F_1 \rightarrow
\]

\[
K=\{\text{"4} \bar{\text{4} \bar{\text{4} \bar{\text{4}}} \text{"}} = \{0X34, 0XB, 0XF6, 0XEA\}
\]

\[
\begin{array}{ccccccccccccccccccccccccccccccccc}
3 & 4 & \text{B} & \text{B} & \text{F} & \text{G} & \text{E} & \text{A} \\
\hline
k_1 & k_1 & k_1 & k_1 & k_1 & k_1 & k_1 & k_1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

\[
\mathbf{q}_1 \rightarrow
\]

\[
C=\text{"0} \bar{\text{0} \bar{\text{0} \bar{\text{0}}} \text{"} = \{0X94, 0XD0, 0XB8, 0XOC\}
\]

\[
\begin{array}{ccccccccccccccccccccccccccccccccc}
9 & 4 & \text{D} & \text{D} & \text{B} & \text{B} & \text{C} \\
\hline
s_{19} & s_{14} & s_{13} & s_{12} & s_{11} & s_{10} & s_{9} & s_{8} \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

\[
\mathbf{q}_2 \rightarrow
\]

Fig.3 Schematic diagram of algorithm encryption

5. Conclusion
This paper presents an improved stream cipher algorithm based on NLFSR, which is consistent with the common stream cipher in time and space complexity. The important difference from the common one is that the security intensity of this algorithm does not completely depend on the key stream, and the uniformity of the key stream is not strictly required so that it has higher security. In addition, this algorithm scrambles bits of plaintext stream and key stream, so it is also applicable to short information.

Acknowledgments
This work was financially supported by the Premier Key-Discipline Enhancement Scheme Supported by Guangdong Government Funds (Grant No. 2016GDYSZDXXK036), And Zhuhai Laboratory of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education And the Foundation of "Three Levels" Backbone Teachers Team Construction Projects of Zhuhai College of Jilin University.

References
[1] Yang Y, Zeng X, Xu Y. Periods on the Cascade Connection of an LFSR and an NFSR[J]. Chinese Journal of Electronics, 2019.
[2] Karunamurthi S, Natarajan V K. VLSI implementation of reversible logic gates cryptography with LFSR key[J]. Microprocessors and Microsystems, 2019, 69(SEP.):68-78.
[3] Caroline Fontaine. Clock-Controlled Generator[M]. Springer US, 2011.
[4] Wenjun Tang. Design of a Stream Cipher Algorithm Based on Latin Square and Time-Varying Symbolic Chaotic System[J], 2020.
[5] Perez-Resa A, Garcia-Bosque M, Sanchez-Azqueta C, et al. Chaotic Encryption for 10-Gb Ethernet Optical Links[J]. IEEE transactions on circuits and systems. I, Regular papers, 2019, 66(2):859-868.
[6] Courtois N T, Meier W. Algebraic attacks on stream ciphers with linear feedback[J]. Proc of Eurocrypt, 2003, 2656:345--359.
[7] N. A. Poluyanenko, Potii A V. Comparison of the volume of array of M-LFSR and M-NLFSR, the generation rate on their basis for GF(2) and in the extensions of the field GF(22)[J]. Telecommunications and Radio Engineering, 2017, 76(12):1101-1110.
[8] Belmeguenai A, Mansouri K, Lashab M. Speech Encryption Using Stream Cipher[D]. US, 2015.