

The Study and Implementation of Correlation Attack on LFSR Based Combination Generators

Raman Preet Singh Khera and L N Das

Abstract - In this paper we discuss the working & implementation of Linear Feedback Shift Registers (LFSRs) in detail and implement the Siegenthaler's correlation attack on a different combination generator with four LFSRs and a nonlinear Boolean function with some defined correlation property. In another section, a correlation between some linear combination of input and output variables to the combining function is determined. The determined correlation is also used in extraction of related information about the correlated input variables. In the simpler version, we assume a correlation implies that the output is equal to one of the input variables with a probability 'p' deviating by a significant amount from 0.5 that is $p > 0.5$ or $p < 0.5$. In the phase-1 of the correlation attack, the occurrence probability of each LFSR output state in the final key-stream output is computed. In the phase-2; a decision mechanism is obtained by observing the fact, a significant deviation of occurrence probability from 0.5 makes the LFSR prone to the correlation attack; whereas a close proximity of occurrence probability to 0.5 that is $p \approx 0.5$ makes an LFSR immune against the correlation attack. The correlation attack implementation is possible only on those LFSRs for which there is a significant correlation between its output state and the output of the combining Boolean function. The deviation will be exploited to retrieve the initial states of the LFSRs which are considered as the secret keys of the crypto-algorithm by counting the number of coincidences between the key-stream and all possible shifts of the output sequence of the LFSRs under consideration until this number agrees with the correlation probability.

Keywords - Correlation Attack, Correlation Probability, Cryptographically Weak System, Geffe Generator, Kerckhoffs's Principle, LFSR, LFSR Based Combination Generator.

1 INTRODUCTION

Correlation attacks are a class of known plaintext attacks which were first proposed on Geffe Generator in 1984 by T. Siegenthaler for breaking a specific class of stream ciphers known as combination generators. In a combination generator the key stream is generated by combining the output of several linear feedback shift registers (LFSRs) using a Boolean function.

Correlation attacks exploit a statistical weakness that arises from an inappropriate choice of the Boolean function. As per the Kerckhoffs's principle (1883) in Cryptography, the security of a cryptosystem should depend solely on the secrecy of the key and the private randomizer. A cryptosystem should be secure even if everything about the system, except the secret key, is public knowledge. Therefore, the details of the cryptosystem is considered to be known to the attacker except for the secret key. Hence in any cryptanalytic attack on a generator, the aim of the attacker is to retrieve the secret key by using the details of the generator and also some portion of the key stream in some cases (known plaintext attack scenario).

LFSRs are the basic components of many running-key generators for stream cipher applications, because they are appropriate to hardware implementation and they produce sequences with good statistical properties. An LFSR can produce a sequence of large period if the feedback polynomial is chosen appropriately. In the next section, we have described with definition how LFSR is implemented.

2 LFSR AND ITS IMPLEMENTATION

2.1 Linear Feedback Shift Registers (LFSRs)

An LFSR is a well-chosen feedback function which can produce a sequence of bits which appears random and which has a very long cycle. Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences.

Definition: A linear feedback shift register (LFSR) of length L consists of L stages (or delay elements) numbered $0, 1, \dots, L-1$, each capable of storing one bit and having one input and one output; and a clock which controls the movement of data.

Functioning of an LFSR: During each unit of time the following operations are performed:

- Raman Preet Singh Khera, B.Tech., Mathematics and Computing Engineering, Department of Applied Mathematics, Delhi Technological University, Delhi-110042, India, PH-+91-9899680688, E-mail: rpsk786@gmail.com
- Laxminarayan Das, Professor, Department of Applied Mathematics, Delhi Technological University, Delhi-110042, India, PH-+91-8826570638, E-mail: Indas@dce.ac.in

- (i) the content of stage 0 is output and forms part of the output sequence;
- (ii) the content of stage i is moved to stage $i-1$ for each i , $1 \leq i \leq L-1$; and
- (iii) the new content of stage $L-1$ is the feedback bit s_j which is calculated by adding together modulo 2 the previous contents of a fixed subset of stages $0,1,\dots,L-1$, which is determined by the feedback polynomial.

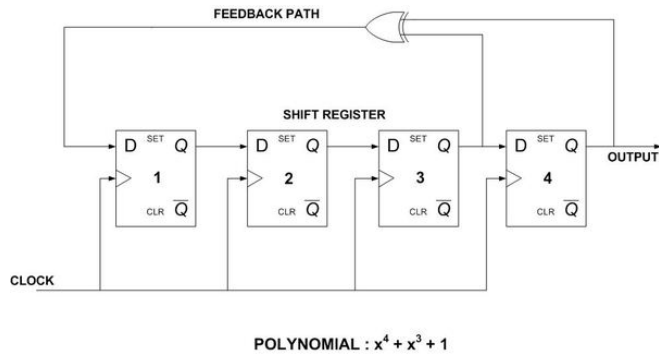


Fig1. Diagram of a 4-stage LFSR

2.2 Implementation of LFSR in C++

We have implemented the functioning of an LFSR in C++. The developed code (namely, `generalizing_lfsr.cpp`) works for an arbitrary length of the LFSR. It asks for input as the length, the feedback polynomial & initial state of the LFSR from the user. It then describes the LFSR states upto a stage when the initial state given by the user is re-obtained & thus gives the period of the LFSR. The maximum period of an LFSR of same length is also mentioned with the output which is given by: $2^L - 1$, where L is the length of the LFSR.

The following is the online link for the developed code; namely `generalizing_lfsr.cpp` uploaded in a repository on GitHub.com :

https://github.com/ramanpreet1993/generalizing_lfsr/blob/master/generalizing_lfsr.cpp

Given below are the snapshots of two test runs of this code with LFSR lengths 4 and 5 respectively. The feedback polynomials are chosen such that the 4-stage LFSR attains its maximum period i.e., 15, while the 5-stage LFSR is of period 21 (i.e., less than the maximum period 31).

TEST RUN - 1

Length of LFSR: 4 Feedback Polynomial : $1+x+x^4$

```

C:\Users\ADMIN\Documents\generalizing_lfsr.exe
Enter the length of LFSR: 4
Enter the feedback polynomial of LFSR :1 0 0 1
Enter the initial state of LFSR at t=0 :1 1 1 1
1: 0 1 1 1
2: 1 0 1 1
3: 0 1 0 1
4: 1 0 1 0
5: 1 1 0 1
6: 0 1 1 0
7: 0 0 1 1
8: 1 0 0 1
9: 0 1 0 0
10: 0 0 1 0
11: 0 0 0 1
12: 1 0 0 0
13: 1 1 0 0
14: 1 1 1 0
15: 1 1 1 1
The period of the lfsr is:15
Maximum period of the lfsr is:15
    
```

Fig2. State-wise description of LFSR having length of 4.

TEST RUN - 2

Length of LFSR : 5 Feedback Polynomial : $1+x^4+x^5$

```

C:\Users\ADMIN\Documents\generalizing_lfsr.exe
Enter the length of LFSR: 5
Enter the feedback polynomial of LFSR :0 0 0 1 1
Enter the initial state of LFSR at t=0 :1 1 1 1 1
1: 0 1 1 1 1
2: 0 0 1 1 1
3: 0 0 0 1 1
4: 0 0 0 0 1
5: 1 0 0 0 0
6: 0 1 0 0 0
7: 0 0 1 0 0
8: 0 0 0 1 0
9: 1 0 0 0 1
10: 1 1 0 0 0
11: 0 1 0 0 0
12: 0 0 1 1 0
13: 1 0 0 1 1
14: 0 1 0 0 1
15: 1 0 1 0 0
16: 0 1 0 1 0
17: 1 0 1 0 1
18: 1 1 0 1 0
19: 1 1 1 0 1
20: 1 1 1 1 0
21: 1 1 1 1 1
The period of the lfsr is:21
Maximum period of the lfsr is:31
Process exited with return value 0
Press any key to continue . . .
    
```

Fig3. State-wise description of LFSR having length of 5.

3 LFSR BASED COMBINATION GENERATOR & ITS IMPLEMENTATION

LFSRs give large period and have very good statistical properties. But their linear complexity is very low. Higher linear complexity is achieved by combining several LFSRs with a non-linear Boolean function.

3.1 Combination Generator based on LFSRs

A system where outputs of several individual LFSR units are involved and combined together as the inputs to a non-

linear Boolean function 'F' to ultimately generate the final key stream bit 'K' is known as an LFSR based combination generator.

The diagram of a Combination generator with n LFSRs is depicted in Fig4.

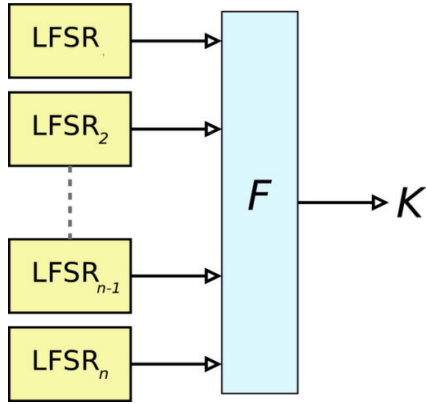


Fig4. Diagram of a Non-linear Combination Generator with n LFSRs

There are many LFSR based combination generators & one such is GEFGE GENERATOR, which is described as follows:

3.2 GEFGE GENERATOR:

It consists of three maximum-length LFSRs whose lengths are: L_1, L_2, L_3 which are pairwise relatively prime, with nonlinear- combining function as:

$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \oplus (1 \oplus x_2) \cdot x_3 = x_1 \cdot x_2 \oplus x_2 \cdot x_3 \oplus x_3$$

The key stream generated has period $(2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$ and linear complexity; $L = L_1 * L_2 + L_2 * L_3 + L_3$.

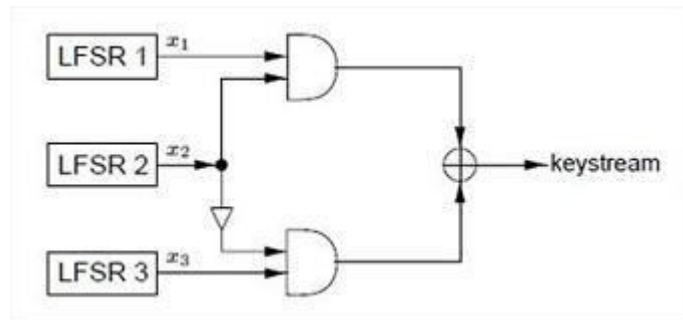


Fig5. Diagram of Geffe Generator

Despite of having large period and high linear complexity, the geffe generator is observed to have some serious cryptographic weakness which is described below:

Let $x_1(t), x_2(t), x_3(t)$ & $z(t)$ denote the t^{th} output bits of LFSRs 1, 2 & 3 & key stream at time 't' respectively.

We know that $z = x_1 * x_2 \oplus x_2 * x_3 \oplus x_3$

$$P(z(t)=x_1(t)) = P(x_2(t) = 1) + P(x_2(t) = 0) \cdot P(x_3(t)=x_1(t))$$

$$= 0.5 + 0.5 * 0.5 = 0.75$$

$$P(z(t)=x_2(t)) = P(x_1(t) \neq x_3(t)) * P(x_3(t) = 0)$$

$$= P(x_1(t) = 1) = 0.5$$

$$P(z(t)=x_3(t)) = P(x_2(t) = 1) + P(x_1(t) = 1) \cdot P(x_2(t)=x_3(t))$$

$$= 0.5 + 0.5 * 0.5 = 0.75$$

It can be seen from the derived probabilities that the information about the initial states of LFSR1 & LFSR3 leaks into the output sequence as the occurrence probability of the two LFSR states in the output sequence deviates from 0.5. Such a system is said to be cryptographically weak and is prone to correlation attack.

4 CORRELATION ATTACK ON COMBINATION GENERATORS

Correlation attacks are possible on combination generators when there is a significant correlation between the output state of one individual LFSR and the output of the combining Boolean function. Suppose that n maximum-length LFSRs R_1, R_2, \dots, R_n of lengths L_1, L_2, \dots, L_n are employed in a nonlinear combination generator. As per the assumptions of Kerckhoffs's principle, if the feedback polynomials of the LFSRs and the combining function f are known, then the number of different keys of the generator that is, total no. of possible initial states (Initial state of an LFSR is considered a secret key) will be

$$\prod_{i=1}^n (2^{L_i} - 1)$$

Let us suppose that there is a correlation between the key stream and the output sequence of R_1 , with correlation probability $p > 0.5$. If a sufficiently long segment of the key stream is known (e.g., as is possible under a known-plaintext attack on a binary additive stream cipher), the

initial state of R_1 can be deduced by counting the number of coincidences between the key stream and all possible shifts of the output sequence of R_1 , until this number agrees with the correlation probability p . Under these conditions, finding the initial state of R_1 will take at most $(2^{L_1} - 1)$ trials. In the case where there is a correlation between the key stream and the output sequences of each of R_1, R_2, \dots, R_n , the (secret) initial state of each LFSR can be determined independently in a total of about

$$\sum_{i=1}^n (2^{L_i} - 1)$$

trials. This number is far smaller than the total number of different keys. In a similar manner, correlations between the output sequences of particular subsets of the LFSRs and the key stream can be exploited.

5 DESCRIPTION OF THE LFSR BASED COMBINATION GENERATOR ON WHICH WE WILL BE IMPLEMENTING THE CORRELATION ATTACK:

The complete description of the generator with input and output details of C++ code is given as follows:

No. of LFSRs = 4

- Length of LFSR1 $l_1 = 7$
- Length of LFSR2 $l_2 = 9$
- Length of LFSR3 $l_3 = 11$
- Length of LFSR4 $l_4 = 13$

Feedback Polynomials:

- For LFSR1 of Degree 7: $1+x+x^7$
- For LFSR2 of Degree 9: $1+x^4+x^9$
- For LFSR3 of Degree 11: $1+x^2+x^{11}$
- For LFSR4 of Degree 13: $1+x+x^3+x^4+x^{13}$

All the four polynomials are chosen to be primitive so that the respective LFSRs generate maximum length sequences.

The non-linear function f which is to be implemented is:

$$f = 1 \oplus (x_1 * x_2 * x_3) \oplus (x_1 * x_3) \oplus (x_1 * x_4) \oplus (x_3 * x_4) \oplus x_1 \oplus x_4 \oplus x_2 \oplus x_3$$

We have to generate key sequence for $t=10,000$ clock cycles.

The following is the figure of the LFSR based combination generator on which we will be applying the correlation attack by first evaluating the occurrence probabilities of the state of every individual LFSR namely s_t^1, s_t^2, s_t^3 & s_t^4 into the final key stream output bit z_t & exploiting the information based on the deviation of occurrence probabilities from 0.5 as per the Kerckhoffs's principle, correlation attack is applied on it.

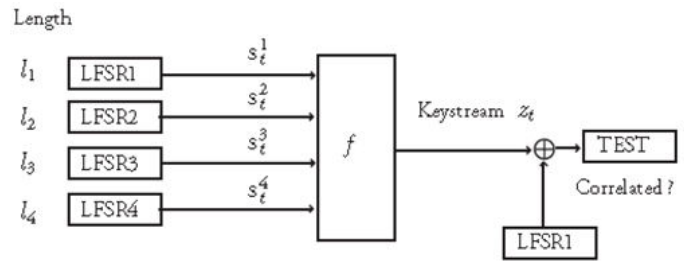


Fig6. Combination Generator comprising of 4 LFSRs along with the non-linear Boolean function 'f' on which Correlation attack will be implemented.

Input : The input to the given C++ code is read from inp.txt data file which contains the no. of LFSRs, length & initial state of each LFSR(considered as the secret key here) which would be derived later on exploiting the information given by the correlation attack and no. of key stream bits to be generated is as given below:

```

4
7
1001010
9
111001001
11
10011000101
13
0100011001111
10000
    
```

The following is the online link for the developed code; namely finding_occurrence_probability.cpp uploaded in a repository on GitHub.com :

https://github.com/ramanpreet1993/finding_occurrence_probability.cpp/blob/master/finding_occurrence_probability.cpp

The C++ code (finding_occurrence_probability.cpp) for the combination generator written above computes the occurrence probability of each LFSR state in the output sequence (and the no. of LFSR bits matched with 10000 key stream bits). It also computes the period of each LFSR. Apart from this, 5 output data files namely: lfsr_seq1.txt, lfsr_seq2.txt, lfsr_seq3.txt, lfsr_seq4.txt & key_seq.txt, each containing 10000 bits are also generated. Below is the output so generated.

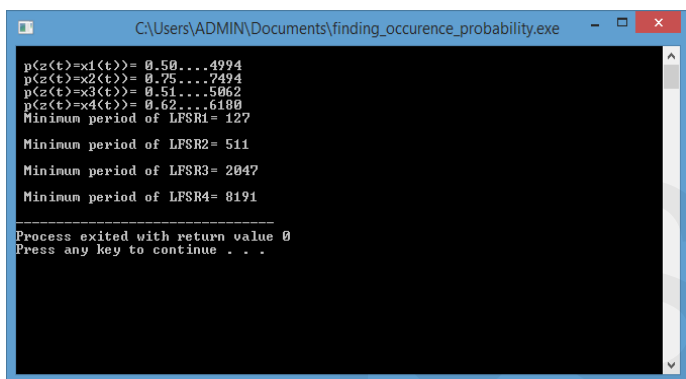


Fig7. Depiction of the occurrence probability of the state of every individual LFSR in the final key stream output bit & the period of every individual LFSR.

OBSERVATION:

Based on the output of the program, we observe that the occurrence probabilities of LFSRs 2 & 4 in the output sequence are 0.75 and 0.63 respectively i.e., these deviate from 0.5 significantly, so they become cryptographically weak & hence, they are prone to correlation attack, therefore their initial states (secret key of our cryptoalgorithm) will be determined by the correlation attack; whereas the occurrence probabilities of LFSRs 1 & 3 in the output sequence are 0.51 & 0.50, i.e., no significant deviation from 0.5; hence they are immune from the correlation attack, therefore their initial states will be determined by the exhaustive search. Therefore, we will now implement correlation attack on the above-mentioned combination generator to determine its complete initial state (secret key) which is not known to the attacker.

6 IMPLEMENTATION OF CORRELATION ATTACK ON COMBINATION GENERATOR

Following the above-mentioned approach, we have implemented the correlation attack on the combination generator as described in Section ‘5’ in C++ programming language. As the correlation probabilities of outputs of LFSR-2 & LFSR-4 with the generator output (i.e., the key sequence) are 0.75 and 0.63 respectively which are deviated from 0.5, this deviation will be exploited to retrieve the initial states of the two LFSRs. As the correlation probabilities in case of other two LFSRs LFSR-1 & LFSR-3 are nearly equal to 0.5, the initial states of these two LFSRs will be determined using exhaustive search. So the complexity of the correlation attack in this case would be

$$(2^{L_2} - 1) + (2^{L_4} - 1) + (2^{L_1} - 1) (2^{L_3} - 1) = (2^9 - 1) + (2^{13} - 1) + (2^7 - 1) (2^{11} - 1) \approx 2^{18} .$$

It may be noted that the complexity of the brute force attack is

$$(2^7 - 1) (2^9 - 1) (2^{11} - 1) (2^{13} - 1) \approx 2^{7+9+11+13} = 2^{40}$$

The input and output details of the program find_lfsr.cpp are given below:

INPUT:

Following parameters, considered to be available with the intruder, are taken as input to the C++ program:

- Size & Feedback Polynomials of the 4 LFSRs
- Boolean function
- Correlation Probabilities
- Key stream (of fixed length)

The first three parameters are hardcoded in the program. The key stream of length 350 bits only from the data file key_seq.txt generated in Section 5 is used for the attack (though there are 10000 bits in this file.)

The following is the online link for the developed code; namely find_lfsr.cpp uploaded in a repository on GitHub.com :

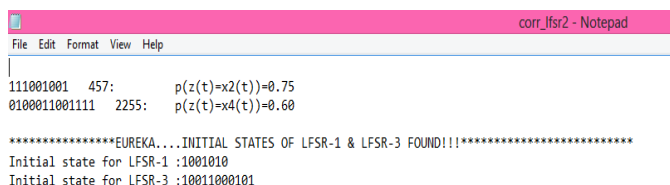
https://github.com/ramanpreet1993/find_lfsr.cpp/blob/master/find_lfsr.cpp

The above written C++ code determines and prints the possible initial states of LFSR-2 & LFSR-4 & also displays

their occurrence probabilities in the output sequence. Then it outputs the initial states of LFSR-1 & LFSR-3.

Thus the complete initial state of the combination generator is found and hence the correlation attack is successfully implemented!!!!

Following snapshot shows the output file corr_lfsr2.txt describing the derived initial states of all four LFSRs. It may be noted that these initial states are same as provided in the input file inp.txt .



```
File Edit Format View Help
corr_lfsr2 - Notepad
111001001 457: p(z(t)=x2(t))=0.75
0100011001111 2255: p(z(t)=x4(t))=0.60

*****EUREKA...INITIAL STATES OF LFSR-1 & LFSR-3 FOUND!!!!*****
Initial state for LFSR-1 :1001010
Initial state for LFSR-3 :10011000101
```

Fig8. Depiction of the derivation of the initial state of every individual LFSR by correlation attack for LFSRs-2 & 4 & by exhaustive search for LFSRs-1 & 3 due to immunity against correlation attack.

7 CONCLUSION

Firstly, we have studied in detail that how a Linear Feedback Shift Register (LFSR) works; we have then successfully implemented its working in C++ programming language. Our developed code works for an arbitrary length of the LFSR. It asks for input as the length, the feedback polynomial & initial state of the LFSR from the user. It then describes the LFSR states up to a stage when the initial state given by the user is re-obtained & thus gives the period of the LFSR. We have tested our developed code for LFSRs of lengths 4 and 5 respectively. The feedback polynomials are chosen such that the 4-stage LFSR attains its maximum period of $2^L - 1$, i.e., 15, while the 5-stage LFSR is of period 21 (i.e., less than the maximum period 31).

Secondly, we have studied and implemented the correlation attack on a combination generator with 4 LFSRs (of lengths 7, 9, 11 and 13). The complete initial state of the generator was recovered using the correlation attack in approximately 2^{18} trials, whereas the brute force complexity to recover the initial state would have been around 2^{40} . The exact complexity of the correlation attack depends on the choice of combining Boolean function. A combination generator can be made immune to the correlation attack by using a suitable Boolean function whose output is not

correlated to any of its input bits. Such Boolean functions are known as correlation immune functions.

8 REFERENCES

[1] A. Menezes, P.van Oorschot and S. Vanstone, Handbook of Applied Cryptography, August, 1996, Chapter-6, Pages: 195-197,203- 208.

[2] T. Siegenthaler, Decrypting a Class of Stream Ciphers using Ciphertext Only, Institute for Communication Technology, Federal Institute of Technology in IEEE Transactions on Computers, Volume 34 Issue 1, January 1985, Pages 81-85, IEEE Computer Society Washington, DC, USA.

[3] T. Siegenthaler, Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications, Published in: IEEE Transactions on Information Theory, Vol.30, Issue: 5, September 1984, Pages: 776-780.

[4] T. Siegenthaler, Correlation attacks on certain stream ciphers with non-linear generators, presented at IEEE Int. Symp. Information Theory, Saint Jovite, Canada, Sept. 26-29, 1983.

[5] J.O. Bruer, on nonlinear combinations of linear shift register sequences, Linkoeeping University Sweden, Internal report March 83, presented at IEEE Int. Symp. Information Theory, Les Arc France, June 21-25, 1982.

[6] Thomas Johansson, Theoretical Analysis of a Correlation Attack Based on Convolutional Codes Member, IEEE, and Fredrik Jönsson, Student Member, IEEE Transactions on Information Theory, Vol.48, No.8, August 2002.

[7] J. Dj. Golic', M. Salmasizadeh, and E. Dawson, "Fast correlation attacks on the summation generator," J. Cryptol., Vol. 13, pp. 245–262, 2000.

[8] W. Meier, and O. Staffelbach, "Fast correlation attacks on stream ciphers", Advances in Cryptology- EUROCRYPT-88, Lecture Notes Computer Science, Vol. 330, Springer- Verlag 1988, pp.301-314.

[9] W. Meier, and O. Staffelbach, "Fast correlation attacks on stream ciphers", Journal of Cryptology, Vol. 1, 1989, pp.159-176.

[10]<https://yorkporc.wordpress.com/2011/04/22/input-output-correlations-in-desfirst-look-at-lfsrand-non-linear-combiners/>