ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 2, No. 4, 2007, *pp*. 288-298

Analysis of Stream Cipher Security Algorithm

Musbah J. Aqel [1 +], Ziad A. Alqadi [2 ++], Ibraheim M. El Emary [3 +++]

[1] Department of Electrical and Computer Engineering,
Faculty of Engineering, Applied Science University, Amman-11931, Jordan
[2] Faculty of Engineering, Al-Balqa Applied University, Amman-Jordan
[3] Faculty of Engineering, Al Ahliyaa Amman University, Amman-Jordan

(*Received October 12, 2006, accepted January 30 2007)*

Abstract. In this paper, the implementation of Berlekamp-Massey algorithm to find the linear complexity for any given sequences is introduced. A new two methods for attacking stream cipher are proposed. The first one is attacking with known combining part using hypothesis test to find the data significant level compromising the appropriate one, while the second method for attacking unknown combining part by finding the behavior (truth table) of the combining part through two algorithms. Once the truth table of the combining part is found, the initial values of the registers can be found in the driving part or drawing the combining part functions using Carnough map or Mcloski algorithm for reverse engineering and reconstruct the combining part.

Keywords: LFSR, Cryptography, Stream Cipher, Cryptanalysis

1. Introduction

Many modern communications and secure systems such as (mobile phone, Bluetooth, SSL, Computer network, etc.) require high speed algorithms to encrypt binary coded plaintext messages which may be several million bits long and display notable structure. The most commonly used cipher in this case is a stream cipher. A stream cipher denotes the process of encryption where binary plaintext is encrypted one bit at a time. The simplest and most often used stream cipher for encrypting binary plaintext is where the bit at a time interval t of a pseudo random sequence zt, is combined using module two addition with the plaintext bit mt, at time interval t to produce the cipher text bit at time interval t, denoted by ct. The sequence zt is called the key stream for the stream. The encryption process can be expressed as:

$$c_t = m_t \oplus Z_t \tag{1.1}$$

Where $\oplus$ denotes module two additions (xor). The decryption process can be expressed as:

$$m_t = c_t \oplus Z_t \tag{1.2}$$

It should be noted, as indicated by equations (1) and (2); that both the encryptor and decryptor need to able to generate the same key stream sequence zt. The key K for the cipher is the initial seed to start the generator. Both the encryptor and decryptor need to process this key. A common method for forming the key stream; zt is to apply a nonlinear Boolean function f to the output binary sequences formed by several linear feedback shift registers (LFSR's) whose characteristic polynomials are primitive.

In this paper, we survey some techniques and approaches for stream ciphers analysis and their requirements. The proposed methods of attacks in this paper are provides three different steps, they are:

- Determining the linear complexity of the key stream sequences generated by the proposal key generators in this paper.
- Determining the initial states of driving part LFSRs or key where the combining part is known using

[+] .*E-mail address*: musbahaqel@yahoo.com
[++] *E-mail address*: natalia_maw@yahoo.com
[+++] *E-mail address*: doctor_ebrahim@yahoo.com

cipher text only approach.

- Determining the initial states of driving part LFSRs (key) as well as to determine the combining function of the combining part using cipher text only approach.

2. Cryptography

Cryptography (or cryptology) is a discipline of mathematics and computer science concerned with information security and related issues, particularly encryption and authentication and such applications as access control. Cryptography, as an interdisciplinary subject, draws on several fields. Prior to the early 20th century, cryptography was chiefly concerned with linguistic patterns. Since then, the emphasis has shifted, and cryptography now makes extensive use of mathematics, including topics from information theory, computational complexity, statistics, combinatory, and especially number theory. Cryptography is also a branch of engineering, but an unusual one as it deals with active, intelligent and malevolent opposition.

Cryptography is the main tool used in computer and network security for such things as access control and information confidentiality.

Cryptography finds many applications that touch everyday life: the security of ATM cards, computer passwords, and electronic commerce all depend on cryptography.

2.1. Stream Ciphers

In cryptography, a stream cipher is a symmetric cipher in which the plaintext digits are encrypted one at a time, and in which the transformation of successive digits varies during the encryption [1]. An alternative name is a state cipher, as the encryption of each digit is dependent on the current state. In practice, the digits are typically single bits or bytes.

Stream ciphers represent a different approach to symmetric encryption from block ciphers. Block ciphers operate on large blocks of digits with a fixed, unvarying transformation. This distinction is not always clear-cut: some modes of operation use a block cipher primitive in such a way that it then acts effectively as a stream cipher. Stream ciphers typically execute at a higher speed than block ciphers and have lower hardware complexity.

Systems where the change of state does not depend on the input (plaintext) to the system are called synchronous (in contrast to asynchronous systems). These systems have the property that every plaintext bit is enciphered independently of the others and an error in one bit does not propagate to other parts of the cipher text.

As described in [2] this has two drawbacks: First, it limits the possibility to detect errors when decrypting. Second, an attacker can insert controlled changes to parts of the cipher text and may achieve a wanted modification of the plaintext.

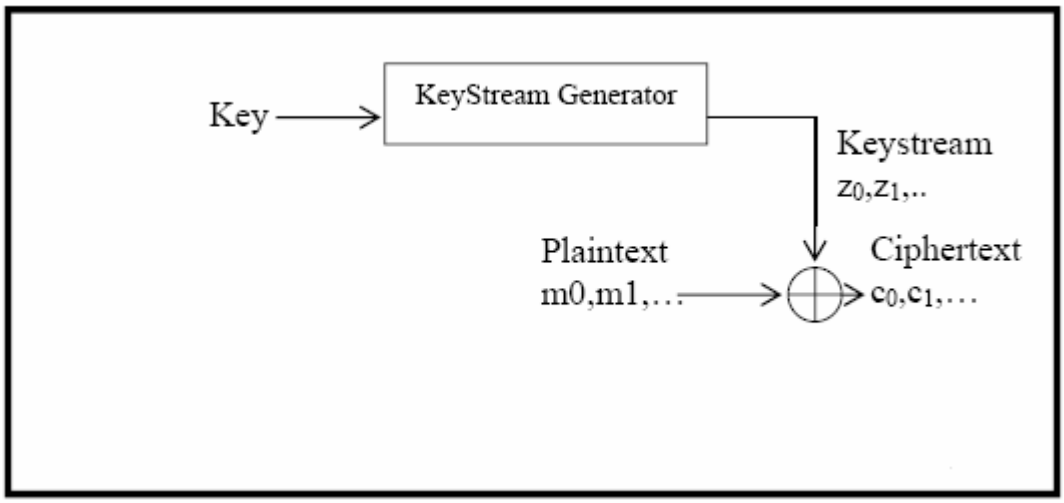

Fig.1 An additive synchronous stream cipher

Thus in the most synchronous stream cipher common form, binary digits are used (bits), and the key stream is combined with the plaintext using the exclusive or operation (XOR).This is termed a binary additive stream cipher, but other functions can also be used. Stream ciphers which use addition as the

combining function as shown in Fig. (1). will be referred to as additive. The sequence produced by the function applied to the internal state is called the key stream. Hereafter only additive synchronous stream ciphers will be discussed.

If we assume that an attacker knows the combining function and is capable of deriving the key stream, the security of a stream cipher depends on whether or not the next character of the key stream can be predicted. There does not seem to be any unified way to determine if a key stream generator produces sequences that are hard to predict. Instead there are numerous tests defined [3] and if a sequence fails any of these tests it is not suitable for use as a key stream. However, a sequence that passes all these tests might yet be vulnerable to some other attack. One important property of a sequence is its period. If used as a key stream it is important that it does not repeat itself during encryption of a plaintext. Thus the period must be longer than the plaintext.

Another test is to use Berlekamp-Massey's algorithm [5] on the sequence to find the shortest linear feedback shift register (LFSR) that can generate the same sequence. The length of this shortest LFSR is called the linear complexity of the sequence.

2.2. Cryptanalysis

Cryptanalysis (from the Greek kryptós, "hidden", and analýein, "to loosen" or "to untie") is the study of methods for obtaining the meaning of encrypted information, without access to the secret information which is normally required to do so. Typically, this involves finding the secret key. In non-technical language, this is the practice of code breaking or cracking the code, although these phrases also have a specialized technical meaning. Cryptanalysis is also used to refer to any attempt to circumvent the security of other types of cryptographic algorithms and protocols in general, and not just encryption. However, cryptanalysis usually excludes attacks that do not primarily target weaknesses in the actual cryptography; methods such as bribery, physical coercion, burglary, key logging, and so forth, although these latter types of attack are an important concern in computer security, and are increasingly becoming more effective than traditional cryptanalysis.

Even though the goal has been the same, the methods and techniques of cryptanalysis have changed drastically through the history of cryptography, adapting to increasing cryptographic complexity, ranging from the pen-and-paper methods of the past, through machines like Enigma in World War II, to the computer-based schemes of the present.

The results of cryptanalysis have also changed it is no longer possible to have unlimited success in code breaking, and there is a hierarchical classification of what constitutes a rare practical attack. In the mid-1970s, a new class of cryptography was introduced: asymmetric cryptography. Methods for breaking these cryptosystems are typically radically different from before, and usually involve solving carefully-constructed problems in pure mathematics, the best-known being integer factorization.

The basic concept of cryptanalysis were developed as a branch of applied mathematics, the cryptanalysis uses the following tools [2]:

- Probability theory and statistics
- Linear algebra
- Abstract algebra (group theory)
- Computer languages
- Complexity theory

One of the most important cryptanalysis tools is a linear complexity of any given sequences, to determine the linear complexity the next section discuss one of the greatest algorithm to calculates the linear complexity.

2.2.1 The Berlekamp Massey Algorithm

The linear complexity of a binary sequence is the length of the shortest LFSR on which the sequence can be generated. For a sequence to be suitable for use as an enciphering sequence in a stream cipher system it is important that it has a sufficiently large linear complexity. There are two forms of linear complexity; global linear complexity, which applies to infinite period binary sequence, and local linear complexity, which applies to binary sequences of finite length.

Consider an n-bit sequence s0 s1 … sn-1.The local linear complexity LC (n) of s0 s1 … sn-1 can be computed using the following Berlekamp-Massey algorithm:

1. $F(x) \leftarrow 1 \quad B(x) \leftarrow 1 \quad D \leftarrow 1$

L ← 0 b← 1 N← 0

2. If N=n, Stop.

Otherwise compute d = $S_N + \sum_{i-1}^{L} C_i S_{N-i}$

1. If d=0, then D ← D+1 and goto(6)
2. if d ≠0 and 2L>N, then F(x) ← F(x)-$db^{-1}x^D$B(x)
   D←D+1 and goto (6)
3. if d ≠0 and 2L≤N, then
   T(x) ← F(x) {temporary storage of F(x)}
   F(x) ←F(x) – $db^{-1}x^D$B(x)
   L←N+1-L
   B(x) ←T(x)
   B ← d
   D ← 1
4. N ← N+1 and return to (2)

Step k requires O (k) operations. Hence the algorithm needs $O(\sum_{k=0}^{L} K) = O(L2)$, operations to analyze a sequence of complexity L.

3. Stream Ciphers Analysis

In this section we introduce the implementation of Berlekamp-Massey algorithm to find the linear complexity for any given sequences, in addition to two new methods for attacking stream cipher the first is attacking with known combining part using hypothesis test to find the data significant level compromising the appropriate one, the second method for attacking unknown combining part by finding the behavior (truth table) of the combining part through two algorithms. Once we find the truth table of the combining part, we can find the initial values of the registers in the driving part or drawing the combining part functions using Carnough map or Mcloski algorithm for reverse engineering and reconstruct the combining part.

3.1. Cipher text only attack for stream cipher using Statistic Methods

This method assumes that the cipher text is afforded, so it is a cipher text only attack. The cipher text is converted into binary form. The Binary sequence of the cipher text is divided into N samples (N>=2) each of which consist of k (k>=2) blocks, for each sample we compute i consecutive 0's preceded and followed by a 1 are called a 0-run of length i. i consecutive 1's preceded and followed by a 0 are called a 1-run of length i.

n0i = # 0-run of length i

n1i = # 1-run of length i

The expected probability of each run:

$$P_r(n_{01})=P_r(n_{11})=\frac{1}{4}=\frac{1}{2^2}$$

$$P_r(n_{02})=P_r(n_{12})=\frac{1}{8}=\frac{1}{2^3}$$

$$Pr(n_{03})=Pr(n_{13})=\frac{1}{16}=\frac{1}{2^4}$$

. . .

$$P_r(n_{0m})=P_r(n_{1m})=\frac{1}{2^{m+1}}$$

It is not difficult to make these distributions and check the hypothesis.

For testing zero hypothesis H0 concerning randomness of binary stream (cipher text stream), it is not sufficient to examine distribution of gaps and blocks in one sample only. It is useful to have information from lot of samples, and use them for making decision. (Sample consists of K blocks containing 2048 bits each. It is recommended $K \geq 2$. Recommended numbers of samples is $N \geq 2$.

System of proposed algorithm consists of the following steps:

- In the created bit stream, distributions of frequencies of blocks and gaps are determined.
- For given number of the blocks and the gaps expected distribution of frequencies of the blocks and the gaps is determined. Since $\chi 2$ criteria are applied, none of expected frequencies should not be less than 10. Because of that, grouping of expected and absolute frequencies is performed. $\chi 2$ distribution with n degrees of freedom is calculated from function [8]:

$$P\left(\chi^2 < \chi_q^2\right) = 1 - \frac{1}{\Gamma\left(\frac{n}{2}\right) \cdot 2^{\frac{n}{2}}} \int_{\chi_q^2}^{\infty} x^{\frac{n}{2}-1} e^{-\frac{x}{2}} dx$$

Where: Γ is a gamma function which is extends the factorial to complex and non-integer numbers (it is already defined on the naturals, and has simple poles at the negative integer). Denoted as [7]:

$$\Gamma(z) = \int_0^{\infty} x^{z-1} e^{-x} dx\,, \quad \forall z > 0$$

When the argument *z* is an integer, the gamma function is just the familiar factorial function, but offset by one,

$$n! = \Gamma(n + 1)$$

The gamma function satisfies the recurrence relation

$$\Gamma(z + 1) = z\Gamma(z)$$

and result of (equation *) checked whether the probability of calculated $\chi 2$ is above the threshold of significance θ. In the same way both distributions (blocks and gaps) are evaluated.

Every $\chi 2$ have $\chi 2$ distribution means that if it is tested N samples, this random value should have $\chi 2$ distribution.

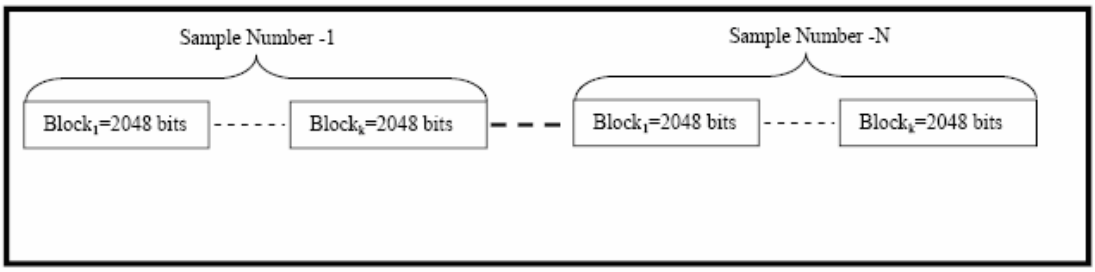

Fig.2 Divide stream cipher text into samples

Algorithm 1.

1. Initialization

N=number of samples; C=-1

M=50; ɵ=0.001 significance threshold for samples

Ɔ=0.00001 significance threshold for all data

Pass0=0; pass1=0

2. Increment C by 1

3. Compute $r = \dfrac{\sum_{i=1}^{m} i(n_{0i} + n_{1i})}{2^2}$

4. Compute the vector $y^{(i)} = \frac{r}{2^i}$
5. Compute $z = \sum_{t=m}^{1} y^{(t)}$ until z≥10
6. d=t-1 (Degree of Freedom)
7. Compute the vector element $y^{(t)}$=z
8. Compute $sn_0 = \sum_{i=d+1}^{m} n_{0i}$ and $sn_1 = \sum_{i=d+1}^{m} n_{1i}$
9. Compute the vector element $S_0^{(d+1)} = Sn_0$ and vector $S_0^{(t)} = n_{0i}$, i=1,…d
10. Compute the vector element $S_1^{(d+1)}$=sn_1 and vector $S_1^{(t)}$=n_1i,i=1.,…d
11. Compute X_2: $X_0 = \sum_{i=1}^{d+1} \frac{(S_0^{(t)} - y^{(t)})^2}{y^{(t)}}$ and $X_1 = \sum_{i=1}^{d+1} \frac{(S_1^{(t)} - y^{(t)})^2}{y^{(t)}}$
12. using equation (*) to compute X_0^{-1} and X_1^{-1} using X_0 and X_1 with d
13. D.O.F as input parameters.
14. if $X_0^{-1} < \text{ө}$ increment pass 0 by 1
    if $X_1^{-1} < \text{ө}$ increment pass1 by 1
15. Compute the vector $n_0^{(j)} = \sum_{i=1}^{m} n_{0j}$ and the vector $n_1^{(j)} = \sum_{i=1}^{m} n_{1j}$,j=1,2,…m
16. go to step (2).
17. Compute $\tau_0 = \sum_{i=1}^{m} in_0^{(i)}$ and $\tau_1 = \sum_{i=1}^{m} in_1^{(i)}$
18. Compute $\lambda = \frac{\tau_0 + \tau_1}{2^2}$
19. Compute the vector $\gamma^{(i)} = \frac{\lambda}{2^i}$,i=1,2,…,m
20. Compute $\eta = \sum_{f-m}^{1} \gamma^{(f)}$ until ή ≥ 10
21. Φ = f -1 (D.O.F)
22. Compute the vector element $\gamma^{(f)} = \eta$
23. Compute the $o_0 = \sum_{i-\Theta+1}^{m} n_0^{(t)}$ and $o_1 = \sum_{i-\Theta+1}^{m} n_1^{(t)}$
24. Compute the vector element $S_0^{(\Theta+1)} = O_0$ and the vector $S_0^{(t)} = O_0^{(t)}$
25. Compute the vector element $S_1^{(\Theta+1)} = O_1$ and the vector $S_1^{(t)} = O_1^{(t)}$
26. Compute $X^2 : X_0 = \sum_{i=1}^{\Theta+1} \frac{(S_0^{(i)} - \gamma^{(i)})^2}{\gamma^{(i)}}$ and $X_1 = \sum_{i=1}^{\Theta+1} \frac{(S_1^{(i)} - \gamma^{(i)})^2}{\gamma^{(i)}}$
27. using equation (*) to compute X_0^{-1} and X_1^{-1} using X_0 and X_1 with D.O.F= as Φ input parameters.
28. if initial generate the properties [(max-pass0 and max-pass1) and (X_0^{-1} < Э) or (X_1^{-1} < Э))] then it is the correct initial and it is solution
29. Stop

As a case study for the implementation of this method a Geffe system is used as a key generator with LFSR's of length (17, 11, and 13) respectively with tapping as shown in Fig.3.

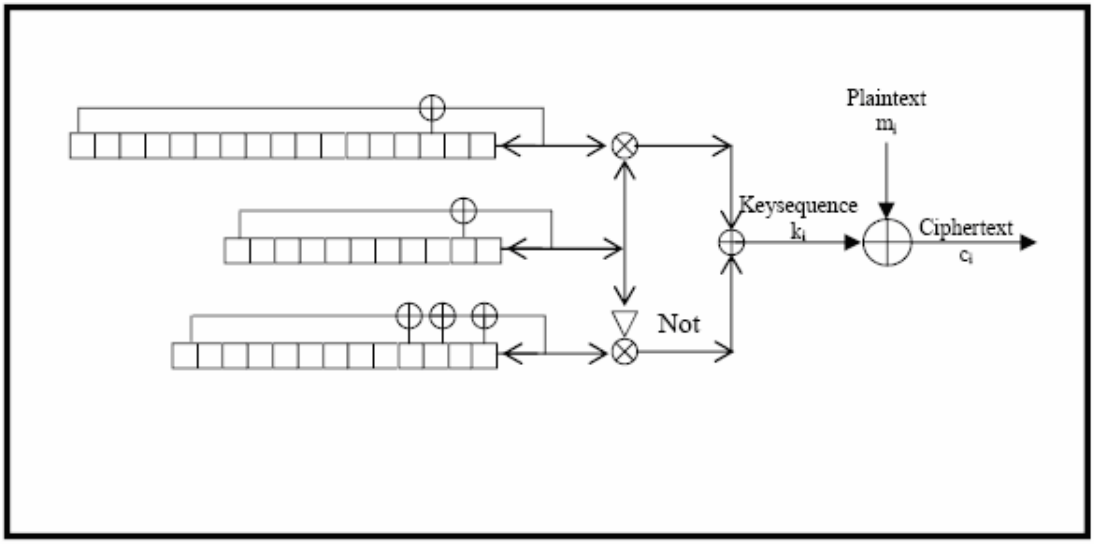

Fig.3 Geffe Stream Cipher System

Table (1) shows the algorithm implementation for each cipher text.

File name	Cipher length (bits)	Maximum number of Blocks	Sample numbers tested	Block numbers per sample tested	Time Complexity (milliseconds)	Plain zeros ratio	Plain ones ratio
C1.txt	16490	8	4	2	4938	0.62	0.38
C2.txt	7985	4	4	1	5175	0.63	0.36
C3.txt	15400	7	7	1	8550	0.61	0.38
C4.txt	21285	10	5	2	8347	0.63	0.36
C5.txt	162725	15	4	2	5925	0.70	0.30
C6.txt	183825	17	4	2	6328	0.70	0.30
C7.txt	91775	8	4	2	6187	0.70	0.30
C8.txt	8910	9	3	3	6772	0.61	0.39
C9.txt	8420	4	2	2	3125	0.63	0.37
CA.txt	12585	6	3	2	4903	0.62	0.38

3.1.1 Attack structure

The attack structure flow chart is explained in Figure (4). The structure can be explained by the following steps:

- Using brute force to find the initial values of the first LFSR, each key sequence generated from distinct initial is mixed (xor) with ciplhertext sequences and the resulted sequence evaluate using algorithm (1).
- Using brute force to find the initial values of the first LFSR, each key sequence generated from distinct initial is mixed (xor) with cipher text sequences and the resulted sequence evaluate using algorithm (1).
- Using brute force to find the initial of the second LFSR (control), using the initial values of the first and third LFSR from above to generate the alternative key sequence the resulted key sequence mixed (xor) with cipher text to compute the 0's percentage of the resulted sequence, the correct plaintext will determine the correct initial.

As we can accomplish the structure of the attack, the divide and conquer method with cipher text only

attack are used to reduce the complexity from search space (2^(17+11+13)=2^41=219902325552 initial system possible to less than or equal search space (2^17+2^11+2^13=141312) initial system possible .

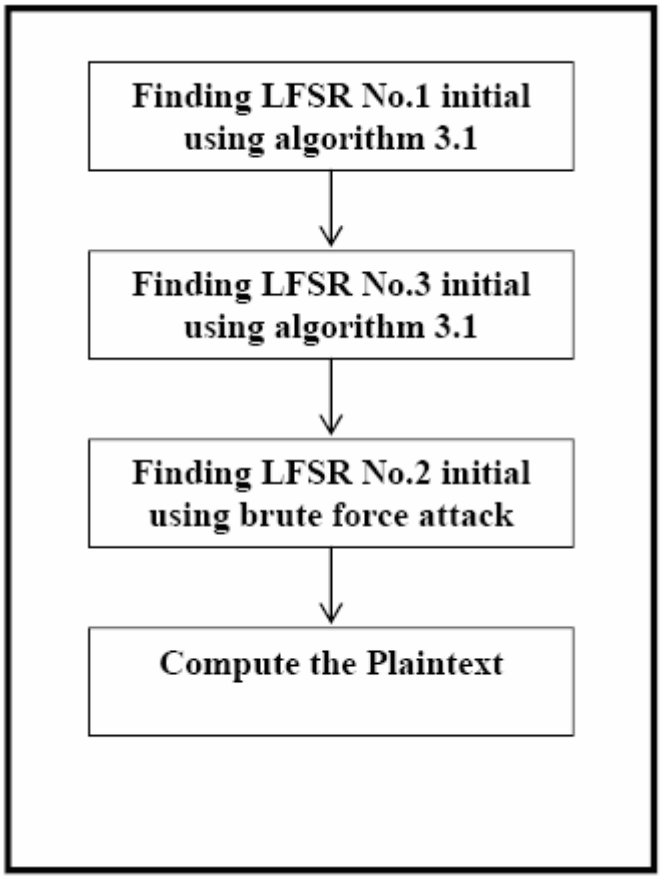

Fig.4 Attack Structure

3.2. Attacking with unknown combining functions

In this section we use this ways stream ciphers for using a proposed techniques. We exploit the proportional calculus to convert any combination of functions to truth table. Besides, we can convert any truth table to non-linear function by using Carnough map or Mcloski algorithm [9].

3.2.1 Proposed Algorithms for Attacking Stream Cipher

In this section we produce a method for attacking stream ciphers algorithms with unknown combining part. To perform this method we have to have the following requirements:

1. The driving part of the stream cipher algorithm. For instance, if we assume that we have an LFSR-based stream cipher, then we have to unknown the number, the length, and the tapping of the LFSR's. The combining part is assumed to be unknown.
2. A cipher text bits of length L. Determining the value of L depending on the number of the output bits of the driving part at each step, such that:

$$L=P*2^n$$

where n is the number of LFSR's in the driving part, and

$$3 \geq P \leq 5$$

For example, if we have 4 LFSR's we need about (48-80) cipher text bits.

In this attack we use brute-force method to produce all possible output of the driving part as an example to show that we can apply this attack on stream cipher, whatsoever is the type of the component of its driving part, besides to study all possible cases that pass the checking of the attack. In this research, we use an LFSR-based stream cipher with three LFSR's of length 5, 6, 7 with tapping explained in Fig.5.

This method of attacking is supposed to attack a key generator with unknown combining part f, so we can assume that the combining part f is a black box of unknown input and output. For this reason we may represent the combining part as a table of 2^n entries. Where n is the number of the driving part registers.

Each input may consider as an address for an empty entry, and the result of the attack is the values that will fill the empty entries. This means that we have to construct the truth table of the combining part, in other words, the attack will yield the behavior of the combining part. One can deduce the structure of the combining part by using K-map or McLoski method.

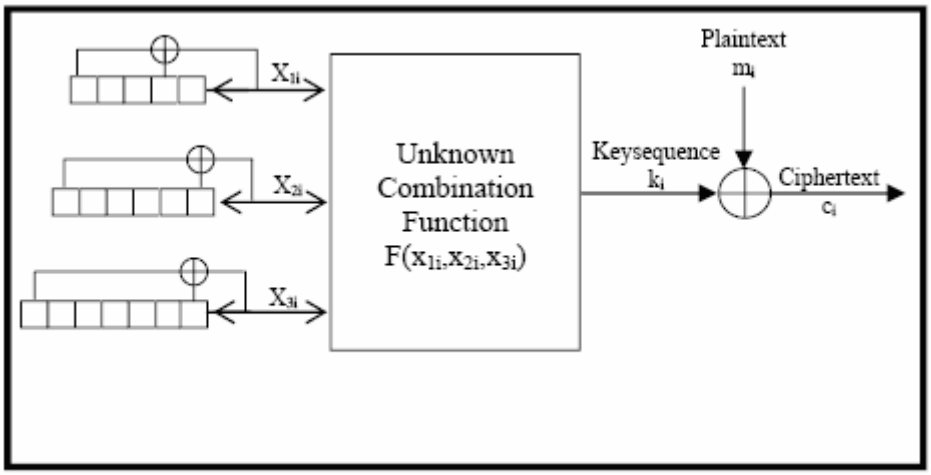

Fig.5 Geffe Stream Cipher System

This method of attacking is supposed to attack a key generator with unknown combining part f, so we can assume that the combining part f is a black box of unknown input and output. For this reason we may represent the combining part as a table of 2^n entries. Where n is the number of the driving part registers. Each input may consider as an address for an empty entry, and the result of the attack is the values that will fill the empty entries. This means that we have to construct the truth table of the combining part, in other words, the attack will yield the behavior of the combining part. One can deduce the structure of the combining part by using K-map or McLoski method.

The attack method can be divided into two main steps, they are:

1. Determining the correct initial state of the driving part.
2. Building the truth table of the combining part.

The first step may be done by producing all possible input and checking the output. In our example we are using brute-force attack which produces all possible initial stages for the registers of the driving part, we do so to test all possible generated states to evaluate the attacking method therefore we selected a relatively short registers but for long registers one can use random search or genetic algorithm techniques.

Table (2) Algorithm implementation parameters

Plaintext Nmae	Plaintext length (bits)	Zero's perc.	One's perc.	Processing time (milliseconds)
P1.TXT	16095	0.62	0.38	8544
P2.TXT	7985	0.63	0.37	4903
P3.TXT	15400	0.61	0.39	7941
P4.TXT	21285	0.63	0.37	10897
P5.TXT	31320	0.62	0.38	16965
P6.TXT	35315	0.63	0.37	18600
P7.TXT	17540	0.60	0.40	9359
P8.TXT	18910	0.60	0.40	9879
P9.TXT	8420	0.61	0.39	4766
PA.TXT	12585	0.61	0.39	7057

Determining the correct initial state can performed by computing the frequencies of zero's and one's of cipher text at certain steps for each input, and computing the highest frequency of the generated addresses.

The plaintext must have the property that the frequency of zero's is greater than the frequency of one's this assumption is derived from testing a large amount of different length plaintext *table(2)* shows a result sample of testing ten sample with different length.

So if the frequencies are equal or likely equal then the initial state of the driving part is dropped. The initial state will be correct if there is a considerably difference between frequencies. In this case if the frequency of zero's is greater than the frequency of one's then the key bit equal to zero otherwise it is equal to one. This fact is derived from the fact that a randomly selected sample will has the feature of the population. The key bit will be the value of the table entry at certain address. This can be done using the following algorithm:

Algorithm (2)

Count=0, find=false, m=p*2n

1. Generate an initial state for the LFSR's
2. Determine the largest frequency address (FL) and the number of its steps
3. For these steps that mentioned in point 4, compute the frequency of zero's (f0) and frequency of one's (f1) of the cipher text bits.
4. if |f0-f1|/f1 ? 0.40 then find=true and go 9
5. increment count by 1
6. if count ≤ m then go to step2
7. stop

Step 6 make use of the fact that the percentage of being the plaintext bit equal to zero pr(p=0)=0.60 for a suitable length of plaintext.

The process of producing the initial state of the driving part is very important to evaluate the correct plaintext. There are two types of brute force attack message exhaustion and key exhaustion. In message exhaustion we obtain the plaintext message which the main target of any attack while in key exhaustion we obtained the secret key which is efficient if the complete ciphering algorithm is known we might use the key to obtain the plaintext message. In the above algorithm we use a key exhaustion brute-force that produces the initial key which is no sufficient because we don't know the combining part of the stream cipher algorithm, so we need to construct the combining part in order to evaluate the plaintext message (the target text).

Once we obtained the correct initial values of the registers which construct the deriving part Building the truth table of the combining part may be done by the following algorithm:

Algorithm (3)

1. a dr=0
2. Compute the frequency of adr among M steps
3. For these steps that mentioned in step2, compute f0 and f1
4. If f0>f1 then table [adr]=0
5. If f1>f0 then table [adr]=1
6. increment adr by 1
7. If adr<2n goto 2
8. Stop

This algorithm didn't guaranteed that the table will be completed, but there will be an empty entries for that cases where f0=f1 (if exist), and this will be filled by decrypting the cipher text and corrected the wrong plaintext bits.

4. Conclusions

It could be concluded the following, as a result of applying the methods discussed in this paper:

- Stream cipher is not recommended to be used for confidentiality because there are many attacks in literatures besides the attacks discussed in this paper, which means that stream cipher is vulnerable.
- To avoid the proposed attacks we recommended that the number of LFSR's in the deriving part must

exceed the ability to perform the algorithms, which means the number of LFSR must be above 20 with nowadays computational capability.

5. References

[1] R. A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer- Verlag, 1986

[2] M. Robshaw. Stream ciphers. *Technical Report TR – 701*. RSALabs, July 1995

[3] G. Carter, and M. H. Hooper, *Randomness Properties of Binary Sequences.* EISS, Karlsruhe, England, 1989

[4] H. Beker, F. Piper. *Cipher Systems, the Protection of Communications*. Northwood Publications, U.K., 1982

[5] J. L. Massey. Shift Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory*, 1969, **T15**(1): 122-127.

[6] W. Henkel. Another Description of the Berlekamp Massey Algorithm. *IEE Proceeding*. 1989, **136**(3).

[7] Y. L. Luke. *Mathematical Functions and Their Approximations*. New York: Academic Press. 1975.

[8] Elliott Mendelson. *Boolean Algebra and Switching Circuits*. McGrraw-Hill Book Co., 1970.