A Fast Correlation Attack on Stream Cipher

Rakesh Shukla

Under the guidance of

Sarbani Palit

CVPR Unit

Indian Statistical Institute Kolkata-700108

Certificate of Approval

This is to certify that the dissertation work entitled A Fast Correlation Attack on Stream Cipher submitted by Rakesh Shukla, in partial fulfillment of the requirements for M.Tech in Computer Science degree of the Indian Statistical Institute, Kolkata, is an acceptable work for the award of the degree.

Date: July , 2007

(Supervisor) (External Examiner)

Acknowledgement

With great pleasure and sense of obligation I express my heartfelt gratitude to my guide and supervisor **Dr.** Sarbani Palit of Computer vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata. I am highly indebted to him for his invaluable guidance and ever ready support. His persisting encouragement, perpetual motivation, everlasting patience and excellent expertise in discussions, during progress of Project Work, have benefited to an extent, which is beyond expression.

The chain of my gratitude would be definitely incomplete without expressing my gratitude to all my batch mates, for their support and encouragement throughout the entire M.Tech course. Lastly I sincerely thank all my friends and well wishers who helped me directly or indirectly towards the completion of this work.

Rakesh Shukla

mtc0516,

Indian Statistical Institute

Kolkata-700108

A Fast Correlation Attack on Stream Cipher

Rakesh Shukla
Supervisor: Dr. Sarbani Palit
July-2007

1. Introduction

Stream ciphers form an important class of ciphersystems. Their speed over that of block ciphers and less complex hardware circuitry make it advantageous to use stream ciphers in many applications.

In a binary additive stream cipher, the ciphertext is produced by bitwise addition of the plaintext with the key stream, all in binary. The key stream generator is initialized using a secret key. a popular key-stream generator used in stream ciphers consists of several LFSRs combined through a nonlinear boolen function.

Attacks that exploit the similarity between the ciphertext and the LFSR outputs, are termed correlation attack. The nature of the cipher system allows each LFSR to be analysed seperately, thus leading to a divide and conquer strategy. The idea of fast correlation attack, which eliminates the need for an exhaustive search of the LFSR initial conditions was first proposed by Meier and Staffelbach [1]. A number of fast correlation attacks were later proposed. However, fast correlation attacks suffer from one or more of the following drawbacks.

1. The presence of a preprocessing phase of considerable complexity which naturally increases the overall decoding time.

2. An iterative phase which takes time to converge.

3. The assumption of a combining functiion that is not correlation immune and also known to the decrypter.

The algorithm proposed here is free of all these restrictions which is in the sideline of Palit, Roy and Arindom [2].

2. Proposed Algorithm

The algorithm can be outlined as follows :

1. For every bit of the cipher stream, generate as many equations as possible by shifting, squaring etc.. The original LFSR feedback polynomial computes the percentage of relations, say r, satisfied by each bit.

2. Sort the bits in a proportion in which they are satisfying the linear equations.

3. Take those bits which has highest proportion.

4. Express the bits, thus, taken in terms of the initial conditions of LFSR and solve the resultant linear system in order to recover the initial conditions.

Note that the system may not always be solvable in which case that particular combination of bits must be rejected.

Computational Complexity

In step 1, the computation of r for each bit position, requires forming at most $(t+1)(\log_2 \frac{N}{k} + 1)$ linear equations, where k is the length of the LFSR, N is the cipher length and t is number of taps. This is because, the number of polynomial relations drived from the LFSR polynomial is bounded above by $\log_2 \frac{N}{k}$; and corresponding to each such relation (including the generating polynomial), we may form, at best, $(t+1)$ linear equations through shifting. For each equation, we have to compute the product of $(t+1)$ terms. Since there are N bits in all, the total time taken by step 1 is $O((t+1)^2 N \log_2 \frac{N}{k})$.
Step 2 takes $O(N \log_2 N)$ time.
Step 3 takes constant time.
Step 4 takes $O(k^3)$ time.

3. Experimental results

1. For $1 + x + x^4 = 0$, $p = 0.7$, $N = 250$

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

2. For $1 + x^4 + x^7 = 0$, $p = 0.7$, $N = 500$

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

3. For $1 + x^{18} + x^{31} = 0$, $p = 0.6$, $N = 500$, we determined 21 correct bits out of chosen 31 bits.

We have tested the result on LFSR of size upto 32. when p was as low as .60 we were not able to form independent system of linear equations but we determined most of LFSR bits correctly.

4. Conclusion

Algorithm proposed here works well when the number of taps is small. At the same time, it eliminates the need for further iterations. Algorithm doesn't use any threshold, it just sorts the bits in a proportion in which they are satisying the linear equations. When p (correlation probability) is low, then for forming an independent syatem of linear equations, we need cipher length of large size.

References

[1].W. Meier and O. Staffelbach (1989). *Fast correlation attack on certain stream ciphers*, Journal of Cryptology, 1, no.159-176

[2].Sarbani palit ,Bimol Roy and Arindom De. *A fast correlation attack on LFSR based stream ciphers*, Applied Cryptography and Network security, LNCS 2846, 1st Int. conf., ACNS 2003, Kunming, China, October, 2003, Springer Verlag Berlin Heidelberg 2003, pp-331-342.

Contents

Abstract

Stream Cipher models are cryptanalysed using statistical techniques assuming that the detailed architecture of the model (except for the key) and cipher text are available. The work is to estimate the secret key with a "reasonable" computational complexity.

Key Words: Stream ciphers, LFSR, Correlation attack, fast correlation attack

1. Introduction

A *cryptosystem* is a five tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of possible *plaintexts*
2. $\mathcal{C}$ is a finite set of possible *keys*
3. $\mathcal{K}$, the *keyspace*, is a finite set of possible keys
4. For each $K \in \mathcal{K}$, there is an *encryption rule* $e_k \in \mathcal{E}$ and a corresponding *decryption rule* $d_k \in \mathcal{D}$. Each $e_k : \mathcal{P} \to \mathcal{C}$ and $d_k : \mathcal{C} \to \mathcal{P}$ are functions such that $d_k(e_k(x)) = x$ for every plaintext element $x \in \mathcal{P}$.

The main property is property 4. It says that if a plaintext x is encrypted using e_k, and the resulting ciphertext is subsequently decrypted using d_k, then the original plaintext x results.

Cryptanalysis (popularly known as code breaking) is the other side of coin. It is assumed that ciphertext is always available to the attacker and in some cases, some plaintext may also be available. We have considered *ciphertext only attack*.

In section 2, general stream cipher model is described in detail, where ciphertext is produced by bitwise addition of plaintext with the key stream, all in binary. The key stream generator is initialised using a secret key. A popular key stream generator is used in stream ciphers consit of several LFSR's combined through a nonlinear combining function.

The secret key, unknown to the decrypter, is normally chosen to be initial conditions for the LFSR's. the LFSR polynomials are assumed to be known.

The objective of the nonlinear combining function is to destroy the inherent linearity present in LFSR sequences. It enables the key stream to have a large linear complexity in order to prevent linear cryptanalysis. However,

depending on the order of resiliency of the function, there is still some correlation between the ciphertext and the LFSR outputs. Attacks that exploit the similarity between the ciphertext and the LFSR outputs, are termed correlation attack. The first attack on this model was due to siegenthaler [1] which has been described in section 3.1. The idea of a fast correlation attack, which eliminates the need for an exhaustive search of the LFSR initial conditions, was first proposed by Meier and Staffelbach [2], This has been described in section 3.2.

It is important to note, however, that the existing fast correlation attacks suffer from one or more of the following drawbacks [3, 4].

1. The presence of a preprocessing phase of considerable complexity which naurally increases the overall decoding time.

2. An iterative phase which takes time to converge.

3. The assumption of a combining function that is not correlation immune and also known as to the decrypter.

The algorithm which has been proposed here is free of all these restrictions which is in the sideline of Palit, Roy and Arindam [5].

2. The stream cipher system architecture and its components

Figure 1 shows the general form of a popular stream cipher system. The generator G produces a random sequence called the 'keystream' (Y). This is X-ORed (added modulo 2), bit-by-bit with the encoded message called the 'plaintext' (M) to produce the 'ciphertext' (C). For decryption, the same keystream must be X-ORed with the ciphertext (in synchronization with the encryption process) to retrieve the encoded plaintext.

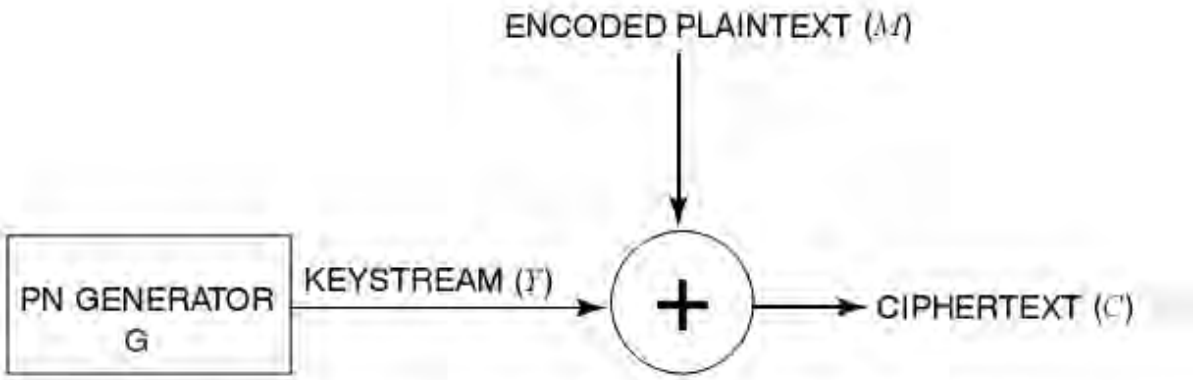

Figure 1. Block diagram of a stream cipher system.

Example: Let M=11010111…… be the plaintext and Y=10010101 …….the computed key stream. Then M is encrypted to C = M ⊕ Y =01000010…… which is sent to the receiver. The legal receiver who can compute Y = 10010101……by himself, decrypts C by M = C ⊕ Y = 11010111…….

A cryptosystem is said to have *perfect secrecy* if $p(x|y) = p(x)\ \forall\ x \in P, y \in C$. P is a finite set of possible plaintexts, C is a finite set of possible ciphertexts and K stands for the set of possible keys. This means that the *a posteriori* probability that the plaintext is x, given that the ciphertext y is observed, is identical to the *a priori* probability that the plaintext is x. Let $e_K \in E$ be the encryption rule and $d_K \in D$ be the decryption rule. Then Shannon provides another charecterization of perfect secrecy suppose :

P, K, C, E, D represents a cryptosystem with $|P| = |K| = |C|$. Then the cryptosystem provides perfect secrecy if and only if every key is used with uniform probability $\frac{1}{|K|}$ and for every $x \in P$ and every $y \in C$ there is a unique key such that $e_K(x) = y$. A well known realization of a perfectly secret system is the Vernam One-time pad. This consists of bit-by-bit X-ORing of the plaintext and keystream to obtain the ciphertext. Decryption is performed by X-ORing the ciphertext and keystream. Most importantly, each key must be used only once which makes the system unconditionally secure and must be of length at least that of the plaintext.

A linear feedback shift register (LFSR) is commonly used to implement a pn sequence. It is both efficient and easy to implement in hardware as well as software. The nth bit of the output generated serially by an LFSR of length d is related to the previous d bits by the linear equation

$$x_i = a_1 x_{i-1} + a_2 x_{i-2} + \cdots + a_d x_{i-d}, \tag{1}$$

$a_1, \ldots, a_d$ being binary constants which, along with the d initial values, characterize the pn sequence. The above equation is often described by means of the polynomial $a(X) = 1 + a_1 X + a_2 X^2 + \cdots + a_d X^d$, known as the feedback or connection polynomial. When the feedback polynomial is *primitive*, i.e., cannot be factorized and any root of it generates the entire field, the period (cycle-length, after which repetition sets in) of the sequence generated is of *maximal length* and equals $2^d - 1$. The longer the period length of the sequence, more is the 'pseudorandomness' of the sequence. Having a long period length is of vital importance to the security of the cryptosystem.

An example of an LFSR with a primitive feedback polynomial $1 + x + x^4$ is shown below in Figure 2. The period of the sequence is $2^4 - 1 = 15$.

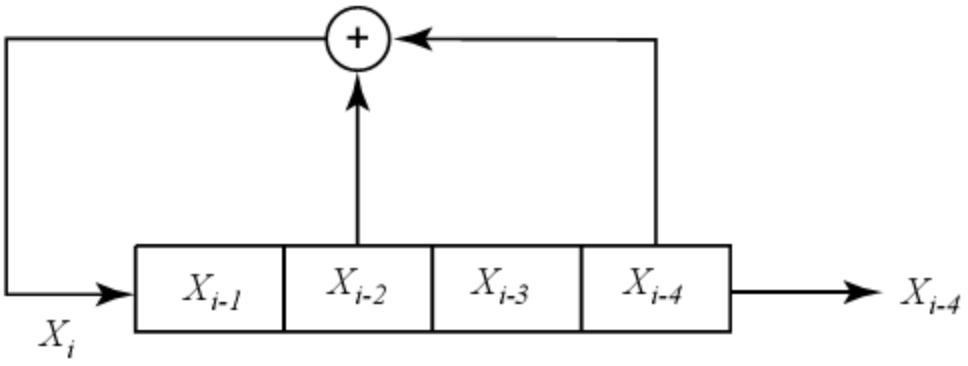

Figure 2. Block diagram of a linear feedback shift register (LFSR) of lenght 4.

Since the bits of the LFSR sequence satisfy a linear recurrence relationship, the use of such a sequence as the keystream leads to an attack of by the Berlekamp-Massey shift register synthesis algorithm

eliminate the possibility of attacks along these lines, the outputs of several LFSRs are combined using a nonlinear Boolean function in order to destroy the inherent linearity present in the keystream. The corresponding system which is one of the most popular stream cipher systems, is shown in Figure 3. The system is initialized with a set of initial conditions for the LFSRs which is the secret *key*.

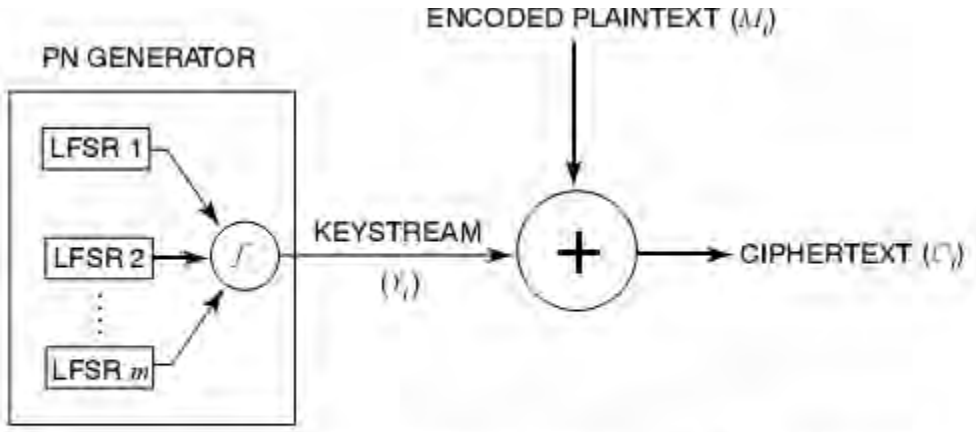

Figure 3. A stream cipher system driven by LFSRs with a combining function.

In such a system, the attacker may face one or more of the following problems, *viz.* unknown initial conditions of the LFSRs, unknown LFSR

polynomials, unknown combining function, availability of limited cipher-length and the need for computation in a reasonable amount of time.

An LFSR of length d_j, has $2^{d_j} - 1$ different choices of the initial conditions. The total number of LFSR initial conditions possible for the system shown in Figure 3 is

$$K = \prod_{j=1}^{m} (2^{d_j} - 1) \text{the all zero condition is never used}$$

If the feedback polynomials of the shift registers are unknown, and R_j is the number of (possibly primitive) feedback polynomials for the jth LFSR then

$$K = \prod_{j=1}^{m} R_j (2^{d_j - 1}).$$

Hence, in a situation when the attacker has access to the ciphertext only, for a *brute force* attack, he must attempt the decryption using all K keys and wait till a meaningful decrypted message is obtained. This is computationally infeasible for LFSRs of even moderate sizes.

3. Sigenthaler's correlation attack

Because of the fact that the probability of 0 in the coded plain text is not exactly equal to half, there is often a non-zero correlation between the keystream and each of the LFSR output sequences. This correlation was first exploited by Siegenthaler to form a divide and conqure approach that the correct initial condition (i.c.) of each LFSR may be determined independently of the i.c.s of the others. He assumed that the shift register sizes and the form of the nonlinear combining function are known. The number of trials to find the i.c.s is then dramatically reduced to

$$\sum_{i=1}^{m} R_i 2^{d_i - 1}.$$

3.1 The method

Let N denote the cipherlength available, $X_1^j, \ldots, X_N^j$ the sequence produced by the jth LFSR, and d_j, the size of the jth LFSR, $j = 1, \ldots, m$. Since each of the LFSR outputs are pn sequences, X_i^j, $i = 1, \cdots, N$ are i.i.d random variables with

$$P_X(X_i^j = 0) = P_X(X_i^j = 1)$$

for all i and j. Further, if the nonlinear combining function f is *balanced, i.e. its output has an equal number of zeros and ones,* then

$$P(Y_i = 0) = P(Y_i = 1).$$

Let $p_0 = P(M_i = 0)$ and $p_j = P(C_i = X_i^j)$. Even though p_j is not directly related to the correlation between C_i and X_i^j, an attack based on the deviation of this probability from one-half is conventionally referred to as a 'correlation attack'. Note that

$$p_j = P(Y_i = X_i^j | M_i = 0)P(M_i = 0) + P(Y_i \neq X_i^j | M_i = 1)P(M_i = 1) \quad (2)$$

Since Y_i and X_i^j are both independent of M_i, we have the simplification

$$p_j = q_j p_0 + (1 - q_j)(1 - p_0) \quad (3)$$

where $q_j = P(Y_i = X_i^j)$.

Consider the random sequence

$$Z_i^j = \begin{cases} 1 & \text{if } C_i = X_i^j, \\ 0 & \text{if } C_i \neq X_i^j. \end{cases}$$

It can be deduced that Z_i^j is a Bernoulli random variable and $\sum_{i=1}^{N}(1-Z_i^j) \sim Bin(N, 1-p_j)$. Thus, for large N, the empirical measure of concurrence between C_i and X_i^j given by

$$\alpha_j = N - 2\sum_{i=1}^{N}(1 - Z_i^j), \quad 1 \leq j \leq m \tag{4}$$

is approximately normally distributed with mean and variance

$$\begin{aligned} m_{\alpha_j} &= N(2p_j - 1), \\ \sigma^2_{\alpha_j} &= 4Np_j(1-p_j). \end{aligned} \tag{5}$$

For all known codes of the plaintext, the probability p_0 is generally different from 0.5. Hence, $p_j = 0.5$ if and only if $q_j = 0.5$. This happens to be the case when the combining function is first order correlation immune i.e., if the function is $f(X_1, X_2, \ldots, X_n)$, then $P(f = X_i) = \frac{1}{2}, \forall i = 1, \ldots, n$.

If p_j is different from 0.5, then α_j is different from 0. On the other hand, if an arbitrary trial sequence is used in place of X_i^j, then $p_j = 0.5$, and consequently

$$\begin{aligned} m_{\alpha_j} &= 0, \\ \sigma^2_{\alpha_j} &= N. \end{aligned} \tag{6}$$

Thus, the question of determining the correctness of a candidate i.c. of the jth LFSR reduces to a test of the null hypothesis $H_0 : \ m_{\alpha_j} = 0$ against the alternative hypothesis $H_1 : \ m_{\alpha_j} \neq 0$. Note that the sequence $X_1^j, \ldots, X_N^j$ needed for computing the test statistic, α_j, is uniquely determined by the candidate i.c., once the connection polynomial is known.

Let us assume, without loss of generality, that $p_j > 0$ for a particular i.c. of the jth LFSR. If the cut-off used for the test statistic α_j is T, then the probability of false alarm is

$$P(\alpha > T|H_0) = 1 - \Phi(T/\sqrt{N}),$$

while the probability of miss is

$$P(\alpha \leq T|H_1) = \Phi((T - (N(2p_j - 1)))/2\sqrt{Np_j(1 - p_j)}),$$

where $\Phi(.)$ is the standard Normal distribution function. Siegenthaler recommends setting the threshold T to ensure a predetermined maximum probability of miss. If several candidate i.c.s exceed the threshold, then all of these should be used to try and decode the ciphertext. If no candidate i.c. is found to exceed the threshold, then a different connection polynomial may be tried out.

3.2 The fast correlation attack

The correlation attacks described so far are based on carrying out an exhaustive search over possible initial conditions. Fast correlation attacks, however, attempt to reconstruct the entire LFSR sequence in an iterative fashion. This section presents the first algorithm of this kind, proposed by Meier and Staffelbach.

3.2.1 The theory

As in the last section we assume that the LFSR sequence is given by (1). The stream cipher system is viewed as a binary noisy channel with the LFSR output at its input. Its output is the ciphertext. The analysis is performed for a single LFSR, though a number of LFSRs can be analyzed similarly. (Consequently we drop the index j.) The channel is assumed to be such that

$$p = P(C_i = X_i) > 0.5 \tag{6}$$

We consider only Boolean functions implying that the coefficients of the polynomial are either 0 or 1. The number of non-zero coefficients a_l, $l = 1, 2, \ldots, d$ give the number of taps or feedback connections. We assume the existence of t such taps. Then, (1) can be rewritten as:

$$\sum_{0 \le l \le d, a_l = 1} X_{i-l} = 0 \tag{7}$$

having $t + 1$ terms. Note that a particular bit, say X_i can be placed in any of the $t + 1$ positions of (1). This implies that X_i simultaneously satisfies $t + 1$ equations of the form (1). Another important observation is

that polynomial multiples of $a(X)$ generate linear relationships satisfied by X and in particular, powers of the form $a(X)^j$, $j = 2^i$, $i = 1, 2, \cdots$, for which $a(X)^j = a(X^j)$. Thus, by repeated shifting of the sequence and 'squaring' of the polynomial, a large number of linear relations with the same number of taps are generated, all of which are satisfied by the bit X_i.

For example, consider the polynomial $1 + x + x^4$ and assume that the cipherlength available is $N = 65$. Then, listed below are the resulting polynomials produced by raising $1 + x + x^4$ to j, where $j = 2^i, \quad i = 1, 2, 3, \cdots$.

$$\begin{aligned}
(1+x+x^4)^2 &= 1 + x^2 + x^8 + 2x + 2x^5 + 2x4(mod2) = 1 + x^2 + x^8 \\
(1+x+x^4)^4 &= 1 + x^4 + x^{16} + 2x^2 + 2x^{10} + 2x^8(mod2) = 1 + x^4 + x^{16} \\
(1+x+x^4)^8 &= 1 + x^4 + x^{16} + 2x^2 + 2x^{10} + 2x^8(mod2) = 1 + x^4 + x^{16} \\
(1+x+x^4)^{16} &= 1 + x^8 + x^{32} + 2x^8 + 2x^{40} + 2x^{32}(mod2) = 1 + x^8 + x^{32} \\
(1+x+x^4)^{32} &= 1 + x^{16} + x^{64} + 2x^{16} + 2x^{80} + 2x^{64}(mod2) = 1 + x^{16} + x^6
\end{aligned}$$

Note that the order of the last polynomial is 64 *i.e.* the corresponding LFSR will have a 64 delay units (the LFSR equation will be: $(X_n = X_{n-16} + X_{n-64})$. Since the length of the data is only 65, generation of any further polynomials by this method will not be of any use. In general, the "squaring" is continued till $2^i d < N$

3.2.2. Underlying Model

The number of linear relations that can be generated for a particular bit $X_i, i = i_1$, say, will naturally be restricted by the cipherlength N available. Each squaring of the polynomial doubles its length and will continue as long as the quantity $2^i d$ is less than N *i.e.* till $i < \lfloor \log_2(\frac{N}{d}) \rfloor$. In other words, the total number of relations obtained

$$T = \sum_{i=0}^{\log_2(N/d)} (N - 2^i d) = N \log_2(\frac{N}{2d}) + d \qquad \textbf{(8)}$$

Since every relation is satisfied by all $t + 1$ bits, the average number of relations per bit equals

$$m = \frac{(t+1)T}{N} \approx \log_2(\frac{N}{2d})(t+1) \qquad \textbf{(9)}$$

Consider the ith bit X_i. These relations may be expressed in the form:

$$L_l = X_i + w_l = 0 \qquad l = 1, \cdots, m, \tag{10}$$

where w_l represents a sum of exactly t different remaining terms with X_i in one of the $t+1$ positions in **(7).**

Consider now a bit of the cipherstream, C_i instead of X_i in (10)

$$L_l = C_i + z_l \qquad l = 1, \cdots, m \tag{11}$$

with z_l representing a sum of exactly t different remaining terms with C_n in one of the $t+1$ positions .

In this case, L_l may not be equal to zero.

Now, let $w_l = w_{l1} + w_{l2} + \cdots + w_{lt}$ and $z_l = z_{l1} + z_{l2} + \cdots + z_{lt}$ where, w_{lj} and z_{lj}, $j = 1, \cdots, t$ are binary variables, all independent and identically distributed with equal probability of being 0 or 1. Note that $P(X_i = C_i) = p = P(w_{lj} = z_{lj})$.

Then, $s(t) = P(w_l = z_l)$ can be recursively computed as follows:

$$\begin{aligned} s(1) &= p \\ s(j) &= ps(j-1) + (1-p)(1-s(j-1)) \qquad j = 2, \cdots, t. \end{aligned} \tag{12}$$

Observe that, for a particular ciphertext bit to satisfy the lth relation *i.e.* $L_l = C_i + z_l = 0$, either $C_i = X_i$ **and** $w_l = z_l$ or $C_i \neq X_i$ **and** $w_l \neq z_l$. Hence

$$\begin{aligned} P(L_1 = \cdots = L_h = 0; L_{h+1} = \cdots = L_m = 1) &= ps^h(1-s)^{m-h} \\ &+ (1-p)(1-s)^h s^{m-h} \end{aligned}$$

where $s = s(t)$. Further

$$
\begin{aligned}
P(C_i = X_i | L_1 = \cdots = L_h = 0; \;\; &= \;\; L_{h+1} = \cdots = L_m = 1) \\
&= \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}} \\
P(C_i \neq X_i | L_1 = \cdots = L_h = 0; \;\; &= \;\; L_{h+1} = \cdots = L_m = 1) \\
&= \frac{(1-p)(1-s)^h s^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}}
\end{aligned}
$$

The basic strategy of the attack is as follows. For a bit C_i, we start with an *a priori* probability $p = P(C_i = X_i) > 0.5$. We count the number h of indices l for which $L_l = 0$. We then alter the *a priori* probability $p = P(C_i = X_i)$ to a new value p^* using **(13)** It is to be expected that

$$\text{if } C_i = X_i \text{ is true}$$

then p^* must increase and vice-versa. This can be verified by computing the expected value of p^* in the two cases.

$$
\begin{aligned}
E(p^* | C_i = X_i) &= \sum_{h=0}^{m} \binom{m}{h} \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}} s^h(1-s)^{m-h} \\
E(p^* | C_i \neq X_i) &= \sum_{h=0}^{m} \binom{m}{h} \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}} s^{m-h}(1-s)^h
\end{aligned}
$$

(13)

3.2.3 The Algorithms

Let

$$\begin{aligned} R &= P(c_n = x_n, \text{and } c_n \text{ satisfies at least } h \text{ of } m \text{ relations}), \\ Q &= P(c_n \text{satisfies at least } h \text{ out of } m \text{ relations}), \\ T &= P(c_n = x_n / c_n \text{satisfies at least h of m relations}). \end{aligned}$$

Then, using (13)

$$Q = \sum_{i=h}^{m} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}) \tag{14}$$

$$R = \sum_{i=h}^{m} \binom{m}{i} ps^i(1-s)^{m-i}, \qquad T = R/Q \tag{15}$$

The quantity h/m which is the minimum fraction of equations that a bit of the cipherstream must satisfy, shall be henceforth, referred to as the upper threshold.

Then, using (13)

$$D = \sum_{i=0}^{h} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}) \tag{16}$$

$$V = \sum_{i=0}^{h} \binom{m}{i} ps^i(1-s)^{m-i}, \tag{17}$$

$$W = \sum_{i=0}^{h} \binom{m}{i} (1-p)s^i(1-s)^{m-i}, \quad E = W/D \tag{18}$$

The maximum fraction of equations h/m, that a bit can satisfy in order to be designated as wrong shall be called the lower threshold. Note that the value of h for the upper threshold is different from that of h for the lower threshold.

Meier and Staffelbach give two algorithms based on these computations. One is an exponential-time attack which is non-iterative in nature. It has limited scope as it has been seen that for $t \geq 10$ and $p \leq 0.75$, this algorithm holds no advantage over an exhaustive search of the initial conditions.

The other algorithm is polynomial time. It starts with a value of h such that the relative increase of correct bits, given by $W - V$ is maximum and a threshold $N = UN$ which is the expected number of bits with $p^* < p_{threshold}$. The value of p^* is calculated from which the no. of bits with p^* less than a threshold, *i.e.* N_w is counted. If this is greater than $N_{threshold}$, only the bits with p^* less than the threshold are complemented and the procedure continued till all the bits equal those of the cipherstream. However, if N_w is less than the threshold, the algorithm must restart with a new *a priori* probability.

It is seen that the polynomial time algorithm stabilizes in only a few iterations.

4. Proposed algorithm

We now propose an algorithm to obtain some bits of LFSR sequence. Once a sufficient number of bits have been correctly determined (slightly more than the length of the LFSR), the initial conditions of the corresponding LFSR are obtained by constructing and solving a system of linear equations.

Note that equation(14) and (15) of last section is generally used for determination of upper threshold and equation(17) and (18) is generally used for lower threshold, but, since the probability of correctly determining the bits increases while the number of bits correctly dtermined decreases. the reverse situation occurs as the lower threshold is increased, the probability that a bit is wrong decraeses while the number of wrong bits increases. Hence, there is a trade-off between ensuring a particular probability of correct determination of bits and obtaining some required number of them.

So, rather than going in above trade-off, we have sorted the bits in a proportion in which they are satisfying the linear equations. We have chosen those bits for solving the system of linear equations which have highest proportion.

The algorithm can be outlined as follows :

1. For every bit of the cipher stream, generate as many equations as possible by shifting, squaring etc.. The original LFSR feedback polynomial computes the percentage of relations, say r, satisfied by each bit.
2. Sort the bits in a proportion in which they are satisfying the linear equations.
3. Take those bits which has highest proportion.

4. Express the bits, thus, taken in terms of the initial conditions of LFSR and solve the resultant linear system in order to recover the initial conditions.

Note that the system may not always be solvable in which case that particular combination of bits must be rejected.

Computational Complexity

In step 1, the computation of r for each bit position, requires forming at most $(t+1)(\log_2 \frac{N}{k} + 1)$ linear equations, where k is the length of the LFSR, N is the cipher length and t is number of taps. This is because, the number of polynomial relations drived from the LFSR polynomial is bounded above by $\log_2 \frac{N}{k}$; and corresponding to each such relation (including the generating polynomial), we may form, at best, $(t+1)$ linear equations through shifting. For each equation, we have to compute the product of $(t+1)$ terms. Since there are N bits in all, the total time taken by step 1 is $O((t+1)^2 N \log_2 \frac{N}{k})$.
Step 2 takes $O(N \log_2 N)$ time.
Step 3 takes constant time.
Step 4 takes $O(k^3)$ time.

Experimental results:

1. For $1 + x + x^4 = 0$, $p = 0.7$, $N = 250$

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

2. The following points should be noted relevant to the explanation of the results:

- An entry of the form $k_1,k_2,\ldots\ldots,k_n$ in the leftmost column of the tables means that, for the LFSR under consideration, the generating polynomial is $1+ D^{k1} + D^{k2} + \ldots + D^{kn}$
- The three sub columns on the right correspond to values of incorrectly determined LFSR bits over chosen bits after sorting for cipher texts of size100, 1000 and 5000.

p=0.7

Polynomial	N=100	N=1000	N=5000
1,22,31,32	10/32	5/32	3/32
2,8,31	2/31	1/31	2/31
18,31	8/31	1/31	0/31
18,25	6/25	1/25	0/25

p=0.8

Polynomial	N=100	N=1000	N=5000
1,22,31,32	7/32	3/32	2/32
2,8,31	1/31	0/31	0/31
18,31	6/31	0/31	0/31
18,25	1/25	0/25	0/25

5. Conclusion

Algorithm proposed here works well when the number of taps is small. At the same time it eliminates the need for further iterations. Algorithm does not use any threshold it just sorts the bits in a proportion in which they are satisfying the linear equations. When the p is low then for forming an independent system of linear equations we need cipher length of large size.

6. References

[1]. T.Siegenthaler,(1985).”Decrypting a class of stream ciphers using cipher text only”, IEEE Transaction on Computers,c-34,N0. 1, 81-85

[2].W. Meier and O. Staffelbach (1989). Fast correlation attack on certain stream ciphers, Journal of Cryptology, 1, no.159-176

[3]. V.V.Chepyzhov, T.Johansson and B.Smeets, “A simple algorithm for fast correlation attack s on stream ciphers,” Fast Software Encryption, 2000.

[4].T.Johansson and F.Jonsson,” Fast correlation attacks based on Turbo Code techniques,” Proceedings of Cryptology-Crypto’99, Springer Verlag, LNCS 1666, pp.181-197.

[5]. Sarbani Palit, Bimol Roy and Arindom De.”A fast correlation attack on LFSR –based stream ciphers”**,** Applied Cryptography and Network security , LNCS 2846, 1st Int. conf. , ACNS 2003, Kunming, China, October 2003, Springer Verlag, Berlin Heidelberg 2003, pp-331-342