A Fast Correlation Attack for LFSR-Based Stream Ciphers

Sarbani Palit[1], Bimal K. Roy[2], and Arindom De[3]

[1] Computer Vision & Pattern Recognition Unit
sarbanip@isical.ac.in
[2] Applied Statistics Unit
bimal@isical.ac.in
[3] Computer Vision & Pattern Recognition Unit
arindom_de@hotmail.com

Indian Statistical Institute, INDIA

Abstract. This paper describes a novel fast correlation attack of stream ciphers. The salient feature of the algorithm is the absence of any pre-processing or iterative phase, an usual feature of existing fast correlation attacks. The algorithm attempts to identify a number of bits of the original linear feedback shift register (LFSR) output from the received bits of the ciphertext. These are then used to construct a system of linear equations which are subsequently solved to obtain the initial conditions. The algorithm is found to perform well for LFSRs of large sizes but having sparse polynomials. It may be noted that such polynomials have low Hamming weight which is one more than the number of feedback connections or "taps" of the corresponding LFSR. Its performance is good in situations even where limited cipherlength is available. Another important contribution of the paper is a modification of the approach when the LFSR outputs are combined by a function which is correlation immune and perhaps, unknown to the decrypter.

Keywords: Stream cipher, Correlation attack, LFSR polynomial, Correlation immune function.

1 Introduction

Stream ciphers form an important class of cipher systems. Their speed over that of block ciphers and less complex hardware circuitry make it advantageous to use stream ciphers in many applications [2]. Such ciphersystems are widely used in military applications across the world [3].

In a binary additive stream cipher, the ciphertext is produced by bit-wise 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 non-linear boolean function. LFSRs lie at the heart of most key-stream generators since they produce pseudo-random sequences with good statistical properties and are easily

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 331–342, 2003.

implemented in hardware. The secret key, unknown to the decrypter, is normally chosen to be the initial conditions of the LFSRs. 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 attacks. The nature of the cipher system allows each LFSR to be analysed separately, thus leading to a divide-and-conquer strategy. Such attacks were first proposed in [14] and further developed in [17,4,6,7,11]. 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 [10]. The algorithm was designed to work for a small number of taps. A number of fast correlation attacks were later proposed in [8,9,5,2,12].

It is important to note, however, that the existing 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 function that is not correlation immune [15] and also known to the decrypter.

The algorithm proposed in this paper is free of all these restrictions. It may be mentioned in this context, that a correlation attack for a situation where the combining function is both correlation immune and unknown was first proposed in [13]. Recovery of the unknown combining function has also been addressed in [3].

Since a divide-and-conquer approach is possible, each LFSR can be analysed independently of the others. In this situation, it is reasonable to model the ciphertext as the output of a binary symmetric channel through which the output of the ith LFSR passes. The error probability of the channel (or the probability that a bit of the ciphertext is not equal to the corresponding bit of the LFSR sequence) is denoted as $(1-p)$. The proposed approach is based on the fact that every bit of an LFSR equation satisfies certain relations called parity checks, all of which are derived from its generating polynomial [10]. Since the cipherstream is correlated to the LFSR output (with probability of concurrence, p, different from .5), many of its bits will also satisfy the same equations. The extent to which a bit satisfies these equations will determine whether it is actually equal to the LFSR bit at the same position. 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 solving linear equations. Section 2 develops the background for the proposed method while the actual algorithm is laid down in Section 3. Performance of the algorithm, comparison

with other well-known algorithms [10,5] are presented in Section 4. Modifications of the algorithm for decryption involving an unknown combining function which may be correlation immune is discussed in Section 5.

2 The Probabilistic Background

The relationships developed in this section (which are used in the present work) have been worked out in much greater detail in [10]. We merely present them here for the sake of clarity and completeness.

Let C denote the ciphertext, N its length available, X_i the sequence produced by the ith LFSR, d_i and t_i the size and number of feedback taps, respectively, of the ith LFSR. With $P(A)$ standing for the probability of the occurrence of the event A, it follows that $p = P(C = X_i)$[1]. Without loss of generality, since only one LFSR will be considered at a time, let $X_i = X$, $d_i = d$ and $t_i = t$. Further, let the LFSR polynomial (assumed to be known) used to generate X, be represented by

$$a(Y) = 1 + a_1 Y + a_2 Y^2 + \cdots + a_d Y^d \tag{1}$$

The number of non-zero coefficients a_i give the number of taps t. Since the sequence $X = \{x_1, x_2, \cdots\}$ is generated from $a(Y)$

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \cdots + a_d x_{n-d} \tag{2}$$

holds with t number of terms on the right hand side. For every bit of X (except those towards the beginning and end of the sequence), placing it in one of the $t + 1$ positions of (2) yields $t + 1$ relations or equations. Now, the number of relations satisfied by each bit can be further increased by noting that every polynomial multiple of $a(Y)$ gives a linear relation satisfied by X. This is true, in particular for powers $a(X)^j$ with $j = 2^i$. It is also important to note that the resulting polynomials all have t feedback taps which is a basic requirement of the algorithm. Hence, repeated squaring (the number of times limited by the cipherlength available) of the LFSR polynomial followed by shifting each bit from one of $t + 1$ positions to another results in a large number of relations all satisfied by the LFSR sequence X.

Let m be the number of relations satisfied by a particular bit x_n (It has been shown in [10] that the average number of relations satisfied per bit is approximately $(t+1)\log_2(N/2d)$). These relations may be expressed in the form:

$$L_i = x_n + w_i = 0 \qquad i = 1, \cdots, m \tag{3}$$

where w_i represents a sum of exactly t different remaining terms with x_n in one of the $t + 1$ positions in (2) and also its multiples.

Consider now a bit of the cipherstream, c_n in place of x_n in (3) with

$$L_i = c_n + z_i \qquad i = 1, \cdots, m \tag{4}$$

[1] Here, the event $C = X_i$ means that a particular bit of C is the same as the corresponding bit of X_i

with z_i representing a sum of exactly t different remaining terms with c_n in one of the $t+1$ positions in (2) and its multiples.

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

Now, let $w_i = w_{i1} + w_{i2} + \cdots + w_{it}$ and $z_i = z_{i1} + z_{i2} + \cdots + z_{it}$ where, w_{ij} and z_{ij}, $j = 1, \cdots, m$ are binary variables, all independent and identically distributed with equal probability of being 0 or 1. Note that $P(x_n = c_n) = p = P(w_{ij} = z_{ij})$.

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

$$\begin{aligned} s(1) &= p \\ s(j) &= ps(j-1) + (1-p)(1-s(j-1)) \qquad j = 2, \cdots, t \end{aligned} \tag{5}$$

Observe that, for a particular ciphertext bit to satisfy the ith relation *i.e.* $L_i = c_n + z_i = 0$, either $c_n = x_n$ **and** $w_i = z_i$ or $c_n \neq x_n$ **and** $w_i \neq z_i$. 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 $s = s(t)$.

Let

$$R = P(c_n = x_n;\ c_n \text{ satisfies at least } h \text{ of } m \text{ relations}),$$

$$Q = P(c_n \text{ satisfies at least } h \text{ out of } m \text{ relations}),$$

$$T = P(c_n = x_n; \text{given that } c_n \text{ satisfies at least } h \text{ of } m \text{ relations}).$$

Then, using (6)

$$R = \sum_{i=h}^{m} \binom{m}{i} ps^i(1-s)^{m-i} \tag{6}$$

$$Q = \sum_{i=h}^{m} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}) \tag{7}$$

$$T = R/Q \tag{8}$$

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.

Further, let

$$W = P(c_n \neq x_n;\ c_n \text{ satisfies at most } h \text{ of } m \text{ relations}),$$

$$D = P(c_n \text{ satisfies at most } h \text{ out of } m \text{ relations}),$$

$$E = P(c_n \neq x_n;\ c_n \text{ satisfies at most h of m relations})$$

Then, using (6)

$$W = \sum_{i=0}^{h} \binom{m}{i} (1-p)s^i(1-s)^{m-i} \tag{9}$$

$$D = \sum_{i=0}^{h} \binom{m}{i} (ps^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i}) \tag{10}$$

$$E = W/D \tag{11}$$

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.

3 Development of the Algorithm

In order to understand the relationship between the various quantities involved in (8),(7),(11),(10), the expressions have been computed for $d = 31$, $k = 2$, $N = 14,000$ and $p = 0.64$. Figure 1a shows the plot of T vs. the upper threshold (varying h/m). Figure 1b shows the plot of the number of bits determined correctly with probability T out of those satisfying the upper threshold (which is equal to $T*Q*N$) vs. the upper threshold Note that satisfying the upper threshold h/m implies satisfying at least h out of m relations. Figure 1c shows the plot of E vs. the lower threshold. Figure 1d shows the plot of the number of bits determined wrongly (and hence to be complemented), vs. the lower threshold.

It can be seen from Figures 1a and 1b, that as the upper threshold is increased, the probability of correctly determining the bits increases while the number of bits correctly determined decreases. The reverse situation occurs in Figures 1c and 1d *i.e.* as the lower threshold is increased, the probability that a bit is wrong decreases while the number of wrong bits increases. It can hence be concluded that the thresholds must be chosen with a trade-off in mind. Not only must the choice ensure a desired probability of correct determination of the bits but at the same time, make it possible that a required number of these are obtained.

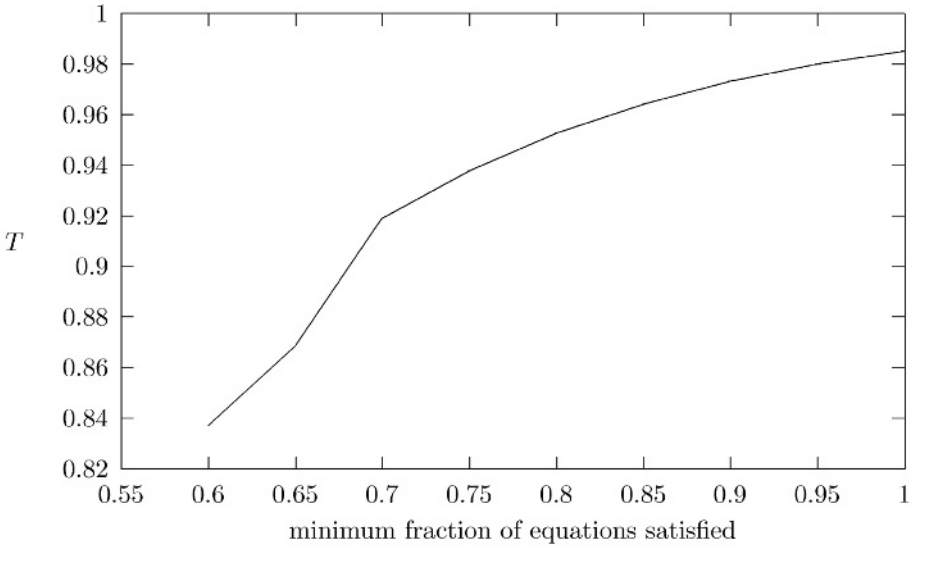

Fig. 1a. T vs. upper threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

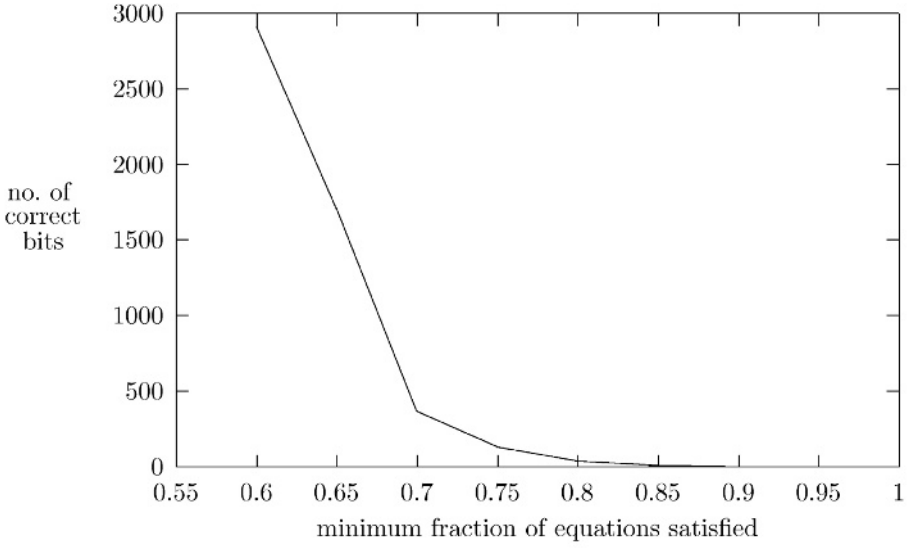

Fig. 1b. No. of correct bits vs. upper threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

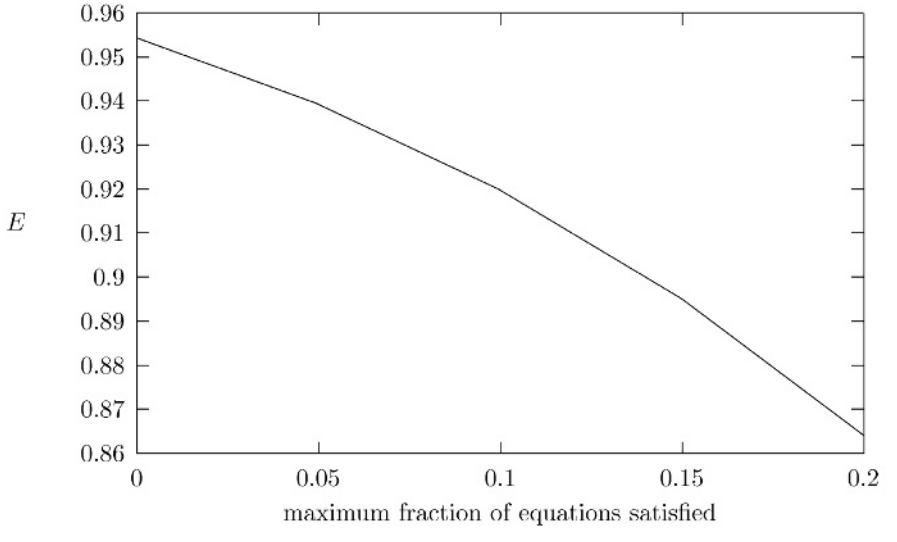

Fig. 1c. E vs. lower threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

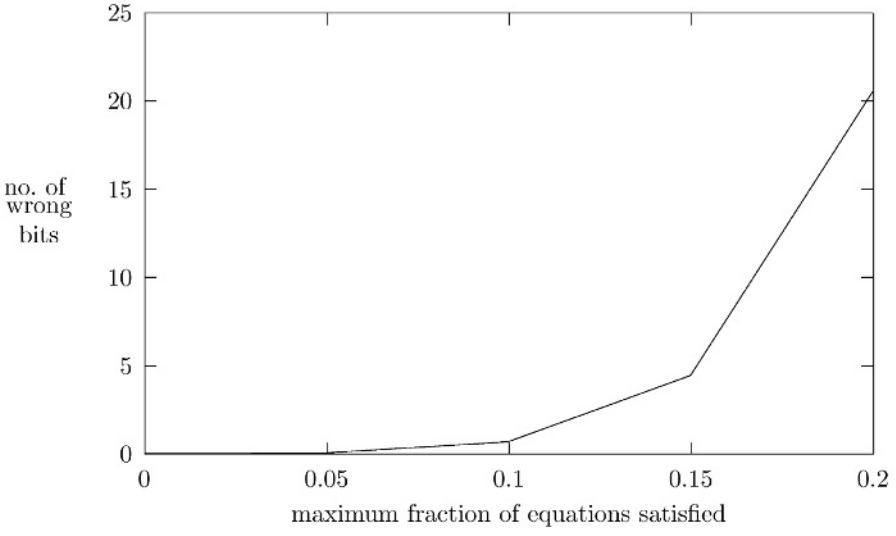

Fig. 1d. No. of wrong bits vs. lower threshold for $d = 31$, $t = 2$, $p = 0.64$, $N = 14000$

The algorithm can now be outlined as follows:

1. For every bit of the cipherstream, generate as many equations as possible by shifting, squaring etc. the original LFSR feedback polynomial. Compute the percentage of relations, say r, satisfied by each bit.
2. Using (8) and (7), obtain the upper threshold such that the probability of correct determination is at least 0.95 and the number of bits correctly determined is at least equal to d, the length of the LFSR.
3. Using (11) and (10), obtain the lower threshold such that the probability of wrongful determination is at least 0.95. Complement these bits.
4. Express the bits thus determined in terms of the initial conditions of the LFSR and solve the resultant linear system in order to recover the initial conditions. Do this for all combinations of the identified bits, taken d at a time. Note that the system may not always be solvable in which case that particular combination of bits must be rejected.
5. From a plot of the frequency distribution of the occurrence of the initial conditions determined in the above step, locate the set/sets occurring most frequently. When this yields only one set, it can be safely assumed that this is the correct initial condition. If there are multiple sets, the Hamming distance between the LFSR output corresponding to each of these and the cipherstream must be computed. The set with the lowest Hamming distance will be the desired initial condition.

4 Performance

Experiments were conducted on LFSRs of various sizes, having up to 6 taps. In each case, the initial conditions were recovered correctly. An illustration of step 4 of the algorithm can be seen in Table 1. It shows the variation of the Hamming distances (normalized by the cipherlength) for the LFSR outputs corresponding to ten candidate sets of initial conditions which occurred most frequently upon solving the linear system of equations as in step 3 for a length 31, 2 tap LFSR with $p = 0.69$. Index no.1 corresponds to the correct initial condition and it clearly has a much lesser Hamming distance than the others.

Table 1. Normalized Hamming distance for various candidate initial conditions.

index	Normalized Hamming distance
1	0.31
2	0.49
3	0.5
4	0.51
5	0.5
6	0.49
7	0.49
8	0.5
9	0.48
10	0.51

It should also be mentioned that the value m used in the theoretical calculations is only an average number and the number of relations for the bits in the central part of the cipherstream is much more. This explains the improved performance of the algorithm in practice. For example, for the 31 length, 2 tap LFSR with $p = 0.69$ considered here, only 4000 cipherbits were sufficient for recovering 36 bits correctly while from theoretical calculations, 14,000 cipherbits appeared to be required. As observed from the graphs of Section 3, the number of bits determined correctly depends on the value selected for the upper threshold, increasing for lower values of the threshold. It actually decreases for higher values of the upper threshold.

Figure 2 shows a plot of $p = P(C = X_i)$ vs. the number of taps for a length 100 LFSR and $N = 20,000$. The value of p is determined using (8) and (7) so that at least 100 bits are correctly determined with a probability greater than 0.95 . Since the contribution from complementing the incorrect bits in this case turned out to be very small, it has been ignored here. As expected, the required correlation between the cipherstream and LFSR sequence increases with the number of taps.

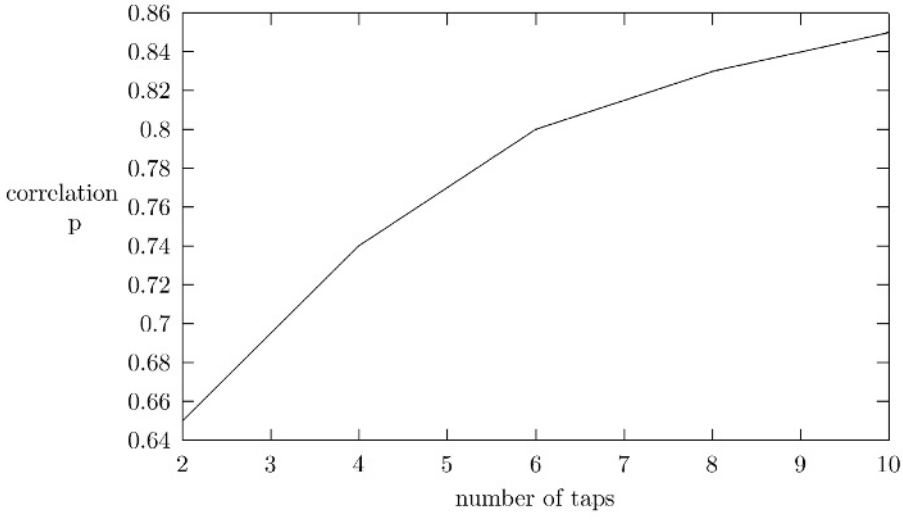

Fig. 2. $p = P(C = X_i)$ vs. no. of taps for $d = 100$, $N = 20000$

From the algorithm, it is obvious that the computational time largely depends on step 4 and consequently, on the number of bits recovered with a certain probability of being correct. Let this number be $d + \Delta$. Then, the maximum number of times that d linear equations have to be solved for the d initial conditions is

$$N_l = \binom{d+\Delta}{d} \tag{12}$$

The choice of Δ thus decides the computation time. A reasonable approach is to go through step 4 with a small value of Δ, say 5% of d. However, if this does not yield a clear solution for the unknown initial conditions, the step must be repeated using a higher value for Δ. It has been observed that higher values are necessary as the number of taps increases. It is also to be noted that a judicious choice of the upper threshold yields a smaller number of bits with a greater chance of being correct, both of which act in the interest of reducing the computational time.

In order to compare the performance of our method with some standard algorithms, some case studies were made. Algorithm B (henceforth referred to as Alg. B) of [10] and Algorithm A1 (henceforth referred to as Alg. A1) of [5] were used for comparison. For the LFSR of length 31, with 2 taps, and $p = 0.69, 0.71, 0.75$, the proposed algorithm and Alg. A1 successfully recover the correct initial condition in each case with a cipherlength of 4000 bits. It was noted, however, that the time taken by Alg. A1 is more than double that of the time taken by the proposed algorithm. Alg. B failed to determine the correct LFSR sequence in each case.

With an LFSR of length 31, 4 taps, $p = 0.75$ and message length of 4000, both the proposed method and Alg. B are successful. Alg. A1 however, fails even with a cipherlength of 10000.

5 Unknown/Correlation-Immune Combining Function

When the combining function is unknown, so is the value of p. Hence the selection of the thresholds, based on the values of T, Q, E and D is no longer directly possible. In such a situation, a search over all possible initial conditions, for the one which has the maximum number of coincidences with the ciphertext, as suggested in [13], works well. Once the initial conditions have been determined, the combining function can be estimated as proposed in [13]. However, the attack can no longer be said to be a fast correlation one since it involves examining all the initial conditions. In such a situation, knowledge of the degree of correlation immunity of the combining function can be appropriately employed.

It has been shown in [16,1] that a Boolean function f is mth-order correlation immune if and only if it is not correlated to linear functions of any subset of m input variables. However, f may be correlated to a linear function of $m+1$ variables, in which case, a correlation attack may be mounted on it. Similarly, in an LFSR based stream cipher system, even though the combining function may not be correlated to the individual LFSR outputs, it may be correlated to certain combinations of them. Hence, it is necessary to determine these LFSR combinations. It is known that a combination of LFSRs (whose outputs are obtained by EX-ORing the individual outputs) may be equivalently replaced by an LFSR whose feedback polynomial is given by the product of constituent LFSR feedback polynomials. For example, if $a(X_i)$ is the generating polynomial of X_i, then $(a(X_1))(a(X_2))$ is the generating polynomial of $(X_1 \oplus X_2)$. If it is known that the combining function is correlated to the output of a particular combination of LFSRs, then it will be correlated to the output of the equivalent LFSR. The algorithm of Section 3 can now be used to determine this output. When the function is unknown, so is the knowledge of such combinations and they must be determined systematically. For instance, if the function is correlation immune of order 1, a combination of two of the LFSRs must be considered. Each of the correctly identified bits is expressed in terms of the initial conditions of either LFSR and hence, both the sets of initial conditions are recovered simultaneously. In order to test this approach, a combining function

$$f = X_3 \oplus X_4 \oplus X_1 X_2$$

was used to combine the LFSRs X_1, X_2, X_3 and X_4. The polynomials used were $(x^{10}+x^3+1)$, (x^9+x^4+1), (x^7+x+1) and (x^6+x^5+1), respectively. Note that $P(f = X_i; i = 1, \cdots, 4) = 0.5$ but $P(f = (X_3 \oplus X_4)) = 0.75$. Hence, the algorithm developed in Section 3 is employed with the corresponding LFSR polynomial $((x^7+x+1) \times (x^6+x^5+1)$. With a cipherlength of 8000, both the sets of initial conditions were recovered successfully.

6 Conclusions

The method presented in this paper appears to perform better than Alg. B of [10] especially when smaller cipherlengths are available and the number of taps is small. At the same time, it eliminates the need for further iterations,

which is an essential feature of Alg. B. It also compares favourably with Alg. A1, being faster and also requiring smaller cipherlength. Our approach does not have a computationally intensive and time consuming pre-processing stage as with other contemporary fast correlation attack algorithms such as Alg. A1. For example, for a polynomial of length 60 and 3 taps and a cipherlength of 14,20,000, the authors of Alg. A1 report a precomputation time of about four days. An important step of the proposed algorithm consists of successively solving a set of linear equation. This set may be made as small as desired, limited only by the length of the LFSR. It may be said therefore that the proposed algorithm requires neither pre-computation nor iterative convergence.

As suggested in the paper, correlation attacks are indeed possible even for unknown combining functions which are correlation immune of a particular degree, provided that sufficient computational power is available. However, the restriction on the number of taps is a bottleneck of such attacks. For the attack on a system with a function which is correlation immune of some order, it should be noted that the generating polynomial of the equivalent LFSR (which is equivalent to the combination of LFSRs, as explained in Section 5) with whose output the cipherlength is correlated, may have a greater number of taps than the constituent LFSRs. However, this does not pose a problem as long as the length of the equivalent LFSR is large enough. In fact, even values of p close to 0.5 may be handled if the LFSR has a very long length and a sufficient amount of ciphertext is available. It may be emphasized again that our observations strengthen the existing views that in order to prevent correlation attacks, the designer needs to choose feedback polynomials with a large number of taps, have a large number of LFSRs and a combining function with a sufficiently high degree of correlation immunity.

References

1. L. Brynielsson, "A short proof of the Xiao-Massey lemma," *IEEE Transactions on Information theory*, vol. IT-35, no.6 (1989), p. 1344.
2. A. Canteaut and M. Trabbia,"Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5," *EUROCRYPT 2000* , LNCS 1807, B. Preneel Ed., Springer-Verlag, Berlin Heidelberg 2000, pp. 573–588.
3. A. Canteaut and E. Filiol, "Ciphertext only reconstruction of stream ciphers based on combination generators," *Fast Software Encryption – FSE 2000*, LNCS 1978, Springer Verlag 2001, pp.165–180.
4. V. Chepyzhov and B. Smeets, "On a fast correlation attack on stream ciphers," *Advances in Cryptology – EUROCRYPT '91*, Vol. 547, D.W. Davies, editor, Springer Verlag, 1991, pp. 176–185.
5. V.V. Chepyzhov, T. Johansson and B. Smeets, "A simple algorithm for fast correlation attacks on stream ciphers," *Fast Software Encryption*, 2000.
6. A. Clark, J. Golić and E. Dawson, "A comparison of fast correlation attacks," *Fast Software Encryption*, Third International Workshop (LNCS 1039), D. Gollman, editor, Springer Verlag, 1996, pp. 145–157.
7. R. Forre, "A fast correlation attack on non-linearly feedforward filtered shift register sequences," *Advances in Cryptology – EUROCRYPT '89* (LNCS 434), 1990, pp. 586–95.

8. T. Johansson and F. Jönsson, "Improved fast correlation attacks on stream ciphers via convolutional codes," *Proceedings of Cryptology – EUROCRYPT '99*, Springer Verlag, LNCS 1592, pp. 347–362.
9. T. Johansson and F. Jönsson, "Fast correlation attacks based on Turbo Code techniques," *Proceedings of Cryptology – Crypto '99*, Springer Verlag, LNCS 1666, pp. 181–197.
10. W. Meier and O. Staffelbach,"Fast correlation attacks on certain stream ciphers," *Journal of Cryptology*, vol.1, no.3, 1989, pp. 159–176.
11. M.J. Mihaljevic and J. Golić, "A comparison of cryptanalytic principles based on iterative error-correction," *Advances in Cryptology – EUROCRYPT '91*, Vol. 547, D.W. Davies, editor, Springer Verlag 1991, pp. 527–531.
12. M. Mihaljević, M. P. C. Fossorier and H. Imai, "Fast Correlation Attack Algorithm with List Decoding and an Application," *Fast Software Encryption- FSE 2000.*
13. S. Palit and B.K. Roy, "Cryptanalysis of LFSR-encrypted codes with unknown combining function," *Advances in Cryptology – ASIACRYPT '99*, Vol. 1716, Springer 1999, pp. 306–320.
14. T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Transactions on Computers*, Vol. c-34, No.1, January 1985, pp. 81–85.
15. T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining functions for Cryptographic Applications," *IEEE Transactions on Information Theory*, Vol. 30, No. 5, September 1984, pp.776–780.
16. G. Xiao and J.L. Massey, " A spectral characterization of correlation-immune combining functions," *IEEE Transactions on Information theory*, vol. IT-34, no.3 (1988), pp. 564–571.
17. K. Zeng and M. Huang, "On the linear syndrome method in cryptanalysis," *Advances in Cryptology – CRYPTO '88*, Vol. 403, S. Goldwasser, editor, Springer Verlag 1990, pp. 469–478.