ISSN:0975-9646
Amar Pandey et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (6) , 2016, 2526-2530

Convolutional Codes based FCA when Combiner Function is 1-CI

Amar Pandey[1], Manoj Kumar Singh[2], N. Rajesh Pillai[3] and S. S. Bedi[4]

[1] *J. K. Institute of Applied Physics & Technology*
Department of Electronics & Communication
University of Allahabad, Allahabad, India

[2, 3, 4] *SAG, DRDO, Metcalfe House, Delhi-54, India*

***Abstract*- Convolution Code based Fast Correlation Attacks (FCA) are a powerful class of attacks on stream ciphers. Thus stream ciphers are designed so that combiner function is correlation immune. In this paper we propose a method to apply Convolution Code based FCA when the combiner function is first order correlation immune (1-CI) but not 2-CI. We apply the attack on a pair of LFSRs whose sum sequence is correlated to key-stream. Instead of directly computing parity checks of equivalent LFSR for the sum sequence, we collect the parity checks of the individual LFSRs into two sets and from the cross product of the set of parity checks, select those pairs in which the two highest degree terms in both the parity checks are identical. These pairs of parity checks are used for setting up the Viterbi decoding phase of the FCA.**

The proposed method is more efficient than the direct FCA using the product of feedback polynomials of LFSRs.

***Keywords*- LFSR, Viterbi decoding, Convolutional code, Correlation attack, Correlation immune, Combiner function.**

I. INTRODUCTION

A popular key-stream generator used in stream ciphers consists of several LFSRs combined through a non-linear Boolean function. However, for certain generators of this type, statistical dependencies or correlations exist between the cipher text and the key-stream sequence produced by an individual LFSR within the key generators. The correlation between two binary segments is a measure of the extent to which they approximate each order. Depending on the order of resiliency of the function, there is still some correlation between the cipher text and the LFSR outputs. Attacks that exploit the similarity between the cipher text and the LFSR outs are termed correlation attacks. The correlation between cipher text and the output of an individual LFSR can be exploited in a divide and conquer attack. Divide and conquer attacks on key stream generators work on each component of the key stream generators separately and sequentially solve for individual initial contents (and possibly, the feedback polynomials as well) of a subset of the input LFSRs from a known segment of the key-stream sequence. These attacks are based on a model where the key-stream is viewed as a noisy version of an underlying LFSR sequence and it is assumed the noise is additive and independent of the underlying LFSR sequence. The attack, if successful, recovers the phase of the LFSR sequence which has the highest correlation with the key-stream sequence. Such attack was first proposed by Siegenthaler [9, 10]. [9] showed that several combining functions previously proposed in the literature can be broken by a cipher text only correlation attack.

For combination generators, the original correlation attack presented by Siegenthaler can be prevented by using a correlation immune combining function [10]. Siegenthaler introduced the concept of m^{th}-order correlation immunity [10] for combining functions as a measure of their resistance against such correlation attacks. He also showed how, by iteration, to construct a limited family of m^{th}-order correlation immune combining functions for every $m,\ 1 \le m \le n$. [15] characterizes all m^{th}-order correlation immune combining functions for every $m,\ 1 \le m \le n$, in terms of their Walsh transforms and extended slightly Siegenthaler's characterization of the algebraic normal form of correlation immune combining functions.

In case the running-key is statistically independent of the output of each constituent LFSR, any correlation attack should then consider several LFSRs together. More generally, a correlation attack on a set of k LFSRs namely LFSR i_1 ……LFSR i_k exploits the existence of a correlation between the running-key s and the output a of a smaller combination generator, which consists of the k involved LFSRs combined by a Boolean function g with k variables. Since

$\Pr[s_n \neq a_n] = \Pr\left[f(X_1, \ldots\ldots, X_n) \neq g\left(X_{i_1}, \ldots\ldots, X_{i_k}\right)\right] \neq p_g$, this attack only succeeds when $p_g < 0.5$. The number k of involved LFSRs should then be strictly greater than the correlation immunity order m of the combining function f. This cryptanalysis therefore requires that all $2^{\sum_{j=1}^{t+1} l_{i_j}}$ initial states be examined, where l_{i_j} is the length of LFSR i_j, it becomes infeasible when the correlation immunity order m of the combining is high. It can be significantly reduced by using some improved algorithms, called fast correlation attacks (FCA).

In this paper we are applying the convolutional code based FCA on first order correlation immune combiner function and giving a new approach for selecting parity check equations and decoding so as to recover initial state of the LFSR.

The paper is organized as follows. In Section II we give overview of Fast Correlation Attack. In Section III we give Convolutional Codes based FCA. In Section IV FCA when combiner function is 1-CI. In Section V we conclude with some possible extensions.

II. FAST CORRELATION ATTACKS- AN OVERVIEW

Fast correlation attacks, as pioneered by Meier and Staffelbach [13, 14], improve on Siegenthaler's correlation attacks by attempting to deduce the initial state without exhaustively trying all possible initial states and considerably reduce the running-time but require a longer segment of known key-stream. Meier and Staffelbach presented two different algorithms A and B for fast correlation attacks, where they use the correlation between the key-stream and the output stream of an LFSR. The algorithms largely depend on the same model of the key generator, the same fundamental observations and the same statistical model. Instead of an exhaustive search over all possible initial states, the algorithms are based on using certain parity check equations created from feedback polynomial of the LFSR. The fast correlation attack algorithm operates in two phases. In the first phase the algorithms find a set of suitable parity check equations based on the underlying LFSR feedback polynomial. In the second phase these parity check equations are applied to the key-stream sequence to determine key-stream bits which, with high probability, are the same as the corresponding bits of the underlying LFSR sequence. A threshold decision process along with an information set decoding technique or an iterative error-correction algorithm is then applied. The algorithm is most efficient when the feedback connection polynomial has only few taps $(t \leq 10)$.

The performance of the algorithms A and B are described in [13, 14]. The algorithms work well when the LFSR contains few taps, but for LFSRs with many taps the algorithms fail. The reason for this failure is that for LFSRs with many taps each parity check equations gives a very small average correlation and hence many equations are needed in order to succeed. In other words, correlation probability p that the algorithms can handle is much lower if the LFSR has many taps $\left(\approx {}^{l}/_{2}\right)$. A new method for finding parity check equations was suggested in [6, 12]. Let a_0 be the initial state of the LFSR. The state after t shifts can be written as $a_t = X^t a_0$, where X is an $l \times l$ matrix that depends of the feedback polynomial. Using powers of the matrix X a set of parity check equations can be found.

More methods for finding parity check equations are suggested in [1, 3, 4, 5, 12]. The underlying idea is to find code words of low weight in a general linear code.

III. CONVOLUTIONAL CODES BASED FCA

Johansson and Jonsson in 1999 introduced the concept of correlation attack using convolutional coding [8, 11]. The parity check equations as described in Fast Correlation Attack designed for a second phase consists of a very simple memory less decoding algorithm. In this approach decoding algorithms are considered to include memory, but still have a low decoding complexity. This work uses the Viterbi algorithm as its decoding algorithm. This algorithm also takes place in two phases. The first phase finds suitable parity check equations that will determine basis of encoder, defining the convolutional code and the second phase includes decoding through Viterbi algorithm. Most decoding algorithms which exploit the structure of the generator matrix use the existence of sparse parity check equations for the linear code C. This technique was first proposed by Meier and Staffelbach in their original paper [13] and later improved [1, 7].

Let the linear code C stemming from the LFSR sequences. There is a corresponding $l \times N$ generator matrix G_{LFSR}, such that $a = a_0 G_{LFSR}$, where a_0 is the initial state of the LFSR. The generator matrix is furthermore written in systematic form i.e. $G_{LFSR} = (I_l, Z)$, where I_l is the $l \times l$ identity matrix. Let B be the fixed memory size and R denote the rate. In convolutional encoder with memory B and rate $R = 1/(m+1)$ the vector v_n of the code word symbols at time n, $v_n = \left(v_n^{(0)}, v_n^{(1)}, \dots\dots, v_n^{(m)}\right)$ is of the form

$$v_n = a_n g_0 + a_{n-1} g_1 + \cdots\dots + a_{n-B} g_B \qquad (1)$$

where each g_i, $0 \leq i \leq B$ is a vector of length $(m+1)$, which is used to encode the information sequence $a = a_0, a_1, \dots\dots\dots, a_N$, i.e. $v = aG$. G will be constructed as the generator matrix of a convolutional code. The parity check equations used in this approach generalize the approach of Meier and Staffelbach in the sense that they use the symbols a_n and (up to) t others $a_i's$, but also any linear combination of the B symbols, a_{n-j}, $0 < j \leq B$, i.e.

$$a_n + \textstyle\sum_{j=1,\dots,B} c_j\, a_{n-j} + \sum_{j=1,\dots,t} a_{n+i_j} = 0 \qquad (2)$$

where the last sum does not necessarily contain t terms. By defining $b_n^{(k)}$, $0 < k \leq m$, as the sum of $t^{(k)} \leq t$ symbols, the complete set of parity check equations can be written as:

$$\begin{aligned} a_n + \textstyle\sum_{j=1,\dots,B} c_j^{(1)}\, a_{n-j} + b_n^{(1)} &= 0, \\ a_n + \textstyle\sum_{j=1,\dots,B} c_j^{(2)}\, a_{n-j} + b_n^{(2)} &= 0, \\ &\dots\dots\dots\dots\dots\dots \\ &\dots\dots\dots\dots\dots\dots \\ a_n + \textstyle\sum_{j=1,\dots,B} c_j^{(m)}\, a_{n-j} + b_n^{(m)} &= 0. \end{aligned} \qquad (3)$$

From (1) & (3), with the addition of the systematic bit $v_n^{(0)} = a_n$, the vectors g_i can be identified as $g_0 = (1,1,\dots\dots,1)$ and $g_i = \left(0, c_i^{(1)}, c_i^{(2)}, \dots\dots, c_i^{(m)}\right)$, $i > 0$. The generator matrix for the convolutional code can then be written as

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_B & & \\ & g_0 & g_1 & g_2 & \cdots & g_B & \\ & & \ddots & & \ddots & \ddots & \ddots \end{pmatrix}$$

To prepare for the decoding, one constructs the received vector r through $r_n^{(0)} = z_n$ and $r_n^{(k)} = \sum_j z_{n+i_j}^{(k)}$, $0 < k \leq m$. The sum over j includes $t^{(k)} \leq t$ terms (cf. (2) and (3)).

The original Viterbi algorithm assumes that the convolutional encoder starts in state 0. But in this application we start from any possible initial state to any ending state for trellis corresponding to the convolutional code. During decoding of a convolutional code, the initial state is usually well defined, but in the current setting, the initial state is unknown. Therefore, the decoding will need to be performed for all 2^B possible initial states. Let

$Z_B = (z_1, z_2, \dots\dots, z_B)$, for each possible starting state $S_B = (s_1, s_2, \dots\dots, s_B)$. Let $d_H(S_B, Z_B)$, the Hamming distance between S_B and Z_B, be initial metric for the state when we start the Viterbi algorithm at $n = B$. Then one runs the Viterbi algorithm over l information symbols. At depth $B + l$ we search for the ending state S_{B+l} with minimum metric. The decoder output is then the information sequence corresponding to the surviving path from one of the starting states S_B to the ending state S_{B+l} with minimum metric. To recover the initial state of the LFSR, it is enough to decode l consecutive information bits correctly. Optimal decoding (ML-decoding) of convolutional codes uses the Viterbi algorithm to decode as follows:

1) For each state S, let $log\left(P\left(S = (z_1, z_2, \dots\dots, z_B)\right)\right)$ be the initial metric for that state when we start the Viterbi algorithm at $n = B$.
2) Decode the received sequence r using the Viterbi algorithm from $n = B$ until $n = B + l$.

Output the estimated information sequence $\left(\widehat{a_{(B+1)}}, \widehat{a_{(B+2)}}, \dots\dots, \widehat{a_{(B+l)}}\right)$. Finally, calculate the corresponding initial state of the LFSR.

In [8], $t = 2$ was used. A variant of the attack using $t = 4$ was proposed by Molland et. al. in [2]. Using $t = 4$ results in many more, but weaker equations. The complexity of this approach is $O(l.2^B.m)$.

IV. FCA WHEN COMBINER FUNCTION IS 1-CI

Assume that the combiner function is first order correlation immune (1-CI); More specifically the output sequence is not correlated to two input sequences b and a but correlated to $b \oplus a$. Let the two sequences b and a be generated by LFSRs with feedback polynomials:

$$f(x) = 1 + f_1 x + f_2 x^2 + \cdots \dots . + f_{l_1} x^{l_1} \text{ and}$$

$$g(x) = 1 + g_1 x + g_2 x^2 + \cdots \dots . + g_{l_2} x^{l_2}$$

of degree l_1 and l_2 respectively. The n^{th} element of the LFSR sequences b_n and a_n can be written as

$$b_n = f_1 b_{n-1} + f_2 b_{n-2} + \cdots + f_{l_1} b_{n-l_1} \text{ and}$$

$$a_n = g_1 a_{n-1} + g_2 a_{n-2} + \cdots + g_{l_2} a_{n-l_2}.$$

To attack such a system using fast correlation attacks, we have to generate equations using the polynomial $f \times g$ which is the feedback polynomial for the equivalent LFSR generating $b \oplus a$. Finding parity equations is a computationally intensive job and is difficult even for l_i about 60, doing it for $deg(f \times g) > 120$ using the usual approach is ruled out.

We first consider the approach of finding parity checks for $f \times g$ by taking products of parity checks of f and parity checks of g. Observe that parity check for a polynomial f is just a multiple of f which satisfies certain additional properties[1]. ([1]Low weight for LDPC decoding or certain distribution of taps for convolution codes). Since product of parity checks of f and g will be divisible by both f and g, we look at the possibility of using them as parity checks.

Fix the parameter B for convolution encoding. We keep the value of this parameter the same for both f and g and find pairs of parity check equations of the form.

$$E_1 : \sum_{i=0\dots B-1} c_i x^i + x^{i_1} + x^{i_2} \quad (4)$$

and

$$E_2 : \sum_{i=0\dots B-1} d_i x^i + x^{i_1} + x^{i_2} \quad (5)$$

where E_1 is a parity check equation for f and E_2 for g; $c_i s$ and $d_i s$ are arbitrary elements of $GF(2)$. Multiplying by equation (4) & (5), we get:

$$E : \sum_{i=0\dots B-1} e_i x^i + x^{2i_1} + x^{2i_2} + 2x^{i_1+i_2} + (x^{i_1} + x^{i_2})(\sum_{i=0\dots B-1}(c_i + d_i)x^i) \quad (6)$$

where E is a parity check equation for $h = fg$, where $e_i s$ are arbitrary elements of $GF(2)$.

If most of the terms of $x^{2i_1} + x^{2i_2} + 2x^{i_1+i_2} + (x^{i_1} + x^{i_2})(\sum_{i=0\dots B-1}(c_i + d_i)x^i)$ get cancelled leaving only t $(t = 2\ or\ 3\ etc)$ terms then we get parity check equations for convolution attacks. But the probability of this happening is very low. Thus this approach has low probability of being applicable.

Here, we give a new approach for selecting parity checks and decoding so as to recover initial state of the LFSR.

In the first step we find all parity check equations of the both polynomials $f(x)$ and $g(x)$ for LFSR as in the original attack. Form pairs (E_1, E_2) consisting of parity check equation of the polynomials $f(x)$ and $g(x)$ such that the degrees of the terms corresponding to last t positions i_1 & i_2 are identical. Only these parity equations will be used in the attack.

The method can now be outlined as follows (Description assumes $t = 2$, but the method is general):

Step-1: For a fixed B, form the parity check equations for f and g, select those parity check equations for which the two highest degree in both LFSR $f(x)$ and $g(x)$ are identical. That is form pairs of parity equations, one from f one from g where last t taps are identical.

Step-2: Find the state tables for convolutional encoder of both LFSR $f(x)$ and $g(x)$.

Step-3: Construct the state table for convolutional encoder of LFSR $h(x)$ with the help of both LFSR $f(x)$ & $g(x)$ state tables. If in a given state S_f, table for LFSR $f(x)$ on input 0 goes to state S_{f0} and gives the output A and on input 1 goes to state S_{f1}, giving output B, and similarly in the state S_g, table for $g(x)$ gives outputs C and D for inputs 0 and 1, going to S_{g0} and S_{g1} respectively. Then in the state table for LFSR $h(x)$ in the current state (S_f, S_g) on input 0 output can either $(A \oplus C)$ with the next state (S_{f0}, S_{g0}) or $(B \oplus D)$ with state (S_{f1}, S_{g1}). Similarly on input 1 output can be either $(A \oplus D)$ with next state (S_{f0}, S_{g1}) or $(B \oplus C)$ with next state (S_{f1}, S_{g0}).

Step-4: Using Viterbi algorithm, we decode so as to find the pair of sequences a' and b' which have the least Hamming distance from $a' \oplus b'$ to the received sequences.

The pseudo codes for the above steps are given in **Algorithm 1**.

Algorithm 1: Algorithm for decoding for product of two LFSRs

procedure MAKE EQUATIONS $(f, g, h, l_1, l_2, l, N, B)$
  $E_1 = \emptyset$ Parity Checks for f
  $E_2 = \emptyset$ Parity Checks for g
  for $i_1 \in B+1 \ldots . N$ **do**
    for $i_2 \in i_1 + 1 \ldots . N$ **do**
      If we get parity check equations for both f & g for i_1 and i_2 **then**
        $E_1 = E_1 \cup (\sum_{i=0 \ldots B-1} c_i x^i + x^{i_1} + x^{i_2})$ Equations for f
        $E_2 = E_2 \cup (\sum_{i=0 \ldots B-1} d_i x^i + x^{i_1} + x^{i_2})$ Equations for g
      end if
    end for
  end for
end procedure
procedure MAKE STATE TRANSITION TABLE (E_1, E_2)
  Design of convolutional encoder for f and g using E_1 and E_2
  T_1 = State table for convolutional encoder for f
  T_2 = State table for convolutional encoder for g
  T = State transition table for h is constructed as follows:
  If $T_1[S_f, 0] = (S_{f0}, A)$ and $T_1[S_f, 1] = (S_{f1}, B)$ and $T_2[S_g, 0] = (S_{g0}, C)$ and $T_2[S_g, 1] = (S_{g1}, D)$ **then**
  $T[(S_f, S_g), 0] = \{((S_{f0}, S_{g0}), A \oplus C), ((S_{f1}, S_{g1}), B \oplus D)\}$
  $T[(S_f, S_g), 1] = \{((S_{f0}, S_{g1}), A \oplus D), ((S_{f1}, S_{g0}), B \oplus C)\}$
  end if Note: Every entry in T has two values instead of one as in T_1 and T_2
end procedure
procedure GENERATE RECEIVED SEQUENCE (z, E_1, E_2, B)
  Given output sequence z_i
  Transform the z_i to r_i for h using $(i_1, i_2)s$ of $E_i s$
end procedure
procedure MODIFIED VITERBI (r, T, B, l)
for each starting state (S_f, S_g), let $d_H(S_f \oplus S_g, z_B)$ be the initial metric
Decode the sequence r using the Viterbi algorithm for $n = B \ldots . B + l$.
For metric use Hamming distance and take min over both the entries of T
Output the estimated information sequence $\left((\widehat{b \oplus a})_{(B+1)}, (\widehat{b \oplus a})_{(B+2)}, \ldots \ldots, (\widehat{b \oplus a})_{(B+l)}\right)$.
Calculate the corresponding initial state of the LFSR $(i.e.\ b \oplus a)$
end procedure

Example: In order to test this new approach, a combining function $f = X_1 \oplus X_2 \oplus X_3 X_4$ was used to combine the LFSRs X_1, X_2, X_3 and X_4. The polynomials used for positions X_1 and X_2 are $x^5 + x^4 + x^3 + x^2 + 1$ and $x^7 + x^5 + x^4 + x^3 + x^2 + x + 1$ respectively. Note that $P(f = X_i : i = 1, \ldots \ldots, 4) = 0.5$ but $P(f = (X_1 \oplus X_2)) = 0.75$. Hence the new method developed in Section IV, is applied with the corresponding LFSR polynomial $(x^5 + x^4 + x^3 + x^2 + 1)(x^7 + x^5 + x^4 + x^3 + x^2 + x + 1)$. With the cipher length *200* of the initial condition for the equivalent register could be recovered.

If we consider the same combining function with the polynomials $x^6 + x^5 + x^4 + x + 1$ and $x^8 + x^6 + x^5 + x^3 + 1$ used at positions X_1 and X_2 respectively. With the cipher length 200 of the initial condition for the equivalent register $(x^6 + x^5 + x^4 + x + 1)(x^8 + x^6 + x^5 + x^3 + 1)$ could be recovered.

V. CONCLUSION

We have given a new method for convolutional codes based fast correlation attack when combiner function is first order correlation immune but not second order correlation immune. The proposed method selects parity check equations of individual LFSRs which have same taps in last t positions and tries to run through the Viterbi algorithm for the LFSRs corresponding to the product LFSR. This is the first time a convolutional codes based fast correlation attack has been applied to a correlation immune function. Possibility of generalization of this method to the case where function is higher order correlation immune is an open area.

ACKNOWLEDGEMENT

The authors wish to thank Dr. P. K. Saxena for continuous support and encouragement for this work. They also wish to thank Dr. Indivar Gupta for their invaluable support and informative suggestions.

REFERENCES

[1] A. Canteaut and M. Trabbia, "Improved fast correlation attacks using parity-check equations of weight 4 and 5", in Advances in Cryptology-EUROCRYPT 2000, ser. LNCS, vol. 1807. Springer-Verlag, 2000, pp. 573588.

[2] Havard Molland, John Erik Mathiassen and Tor Hellasath, "ImprovedFast Correlation Attack Using Low Rate Codes", K. G. Paterson (Ed); Cryptology and coding 2003, LNCS 2898, pp. 67-81,2003.@Springer- Verlag Heidelberg 2003.

[3] M. Mihaljevic, M. P. C. Fossorier and H. Imai, "Fast Correlation Attack Algorithm with List Decoding and an Application", Fast Software Encryption-FSE 2000.

[4] Meier W., "Fast Correlation Attacks: Methods and Countermeasures". In: Joux, A. (eds) Fast Software Encryption 2011. Lecture Notes in Computer Science, pp.5567. Springer-Verlag 2011.

[5] M. Agren, C. Londahl, M. Hell, T. Johansson, " A survey on fast correlation attacks", Department of Electrical and Information Technology, Lund University, Sweden, Springer Science + Business Media, LLC 2012.

[6] M. Mihaljevic and J. Golic, "A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence", Advances in Cryptology-AUSCRYPT-90, Lecture notes in science, vol.453, Springer-Verlag, 1990, pp.165-175.

[7] Sarbani Palit, Bimal K. roy and Arindom De, "A Fast Correlation Attack for LFSR based Stream Cipher", LNCS-2846, pp. 331-342, Springer-Verlag Berlin Heidelberg 2003.

[8] T. Johansson and F. Jonsson, "Improved fast correlation attacks on stream ciphers via convolutional code", In Advances in Cryptology-EUROCRYPT-99, volume 1592 of Lecture Notes in Computer Science. Pages 347-362, Springer, 1999.

[9] T. Siegenthaler, "Decrypting a class of stream ciphers using cipher text only", IEEE Transactions on Computers, Vol.c-34, No. 1, January 1985, pp. 8185.

[10] T. Siegenthaler, "Correlation Immunity of Non linear Combining function for Cryptographic Applications", IEEE Transacations on Information Theory, Vol. 30, No. 5, September 1984, pp. 776780.

[11] T. Johansson and F. Jonsson, "Fast Correlation Attacks based on Turbo Code Techniques", Proceedings of Cryptology Crypto 99, Springer Verlag, LNCS 1666, pp. 181197.

[12] V. V. Chepyzhov, T. Johansson and B. Smeets, "A simple algorithm for fast correlation attacks on stream ciphers", Fast Software Encryption, 2000.

[13] W. Meier and O. Staffelbach, "Fast correlation attacks on certain stream ciphers", Journal of Cryptography, pp. 159-176, 1989.

[14] W. Meier and O. Staffelbach, "Fast correlation attacks on stream ciphers", In Advances in Cryptology- EUROCRYPT 88, volume LNCS 330, pages 301-314, Springer-Verlag, 1988.

[15] Xiao Guo-Zhen and James L. Massey, Fellow, IEEE, "A Spectral Characterization of Correlation Immune Combining Functions", IEEE Transaction Theory, Vol. 34, No. May 1988.

****