International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014)

Correlation Attacks on Stream Cipher

Amar Pandey

J. K. Institute of Applied Physics and Technology, Department of Electronics &Communication University of Allahabad, Allahabad-211002 (U.P.), India

***Abstract*-- Correlation attacks on stream cipher are divide and conquer attacks applied to nonlinear combination generators based on linear feedback shift registers. These are cipher text only attacks that exploit the correlation between the cipher text and the underlying shift register sequences to recover the initial state of the underlying LFSRs. Fast correlation attacks are based on use of parity check equations on which a decoding technique or an iterative error-correction algorithm is applied to recover the initial state. The fast correlation attack using the Viterbi decoding algorithm that applied principles from convolution coding theory gives substantial improvement over previous attacks.**

Keywords-- Stream Cipher, LFSR, Generators, Convolutional Codes, Fast Correlation Attack.

I. INTRODUCTION

In an additive synchronous stream cipher, the cipher text is obtained by adding bitwise the plaintext to a pseudo-random sequence called the key stream. This key stream is generated by a key stream generator (a finite state automation) whose initial state is derived from the secret key, and usually from a public initial value, by a key-loading algorithm. At each time unit, the key stream digit produced by the generator is obtained by applying a filtering function to the current internal state. The internal state is then updated by a transition function. Both filtering function and transition function must be chosen carefully in order to make the underlying cipher secure. In particular, the filtering function must not leak too much information on the internal state and the transition function must guarantee that, for (almost) all initial states, the sequence formed by the successive internal states has a high period [1].

Key stream generators based on LFSRs are commonly used for stream cipher applications, because of the simplicity and speed of the LFSR implementation in hardware and the good key stream properties with regard to period and statistics which sequences produced by LFSRs possess. A key stream is generated by combining the output of a number of LFSRs using a nonlinear combining function f as depicted in Figure 1.

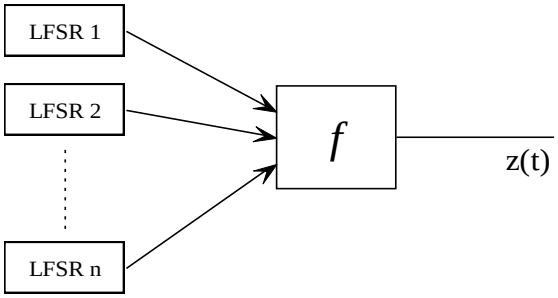

Figure 1. Non linear Combination Generator

However, for certain generator of this type, statistical dependencies exist between the cipher text and the key stream sequence produced by an individual LFSR within the key generator. The correlation between two binary segments is a measure of the extent to which they approximate each order. 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.

The following model is assumed for correlation attack: Assume that we have an observed key stream sequence, z of length, $z = z_1, z_2, \ldots, z_N$ and the connection Polynomial of LFSRs [primitive over $GF(2)$] and the nonlinear combining Boolean Function are known. The key stream sequence is generated from a generator with n different LFSRs. If one can find a correlation between the output of one of the shift registers, called the target LFSR, and the key stream, i.e. $P(a_i = z_i) \neq 0.5$, where a_i is the output of the LFSR and z_i is the known key stream symbol. We have to determine the initial states of LFSRs. For this we discuss the following correlation attacks:

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014)

1. Correlation attack
2. Fast Correlation attack
3.Convolutional Code based Fast Correlation attack

II. Correlation Attack

Correlation attacks had their origin in a paper by Seigenthaler in 1985 described as a cipher text only attack [4, 5]. For the regularly clocked nonlinear combination generator, Siegenthaler showed that the correlation between each of the inputs, $a_i, 1 \le i \le n$, to the combining function and the output, z, can be exploited in a divide and conquer attack. The attack is based on a model where the key stream is viewed as a noisy version of an underlying LFSR sequence, with the noise assumed to be additive, and independent of the underlying LFSR sequence. i.e. $z(t) = a(t) \oplus e(t)$, for $t > 0$, where $z(t), a(t)$ and $e(t)$ denote the i^{th} bits in the key stream sequence, the i^{th} LFSR sequence and the noise sequence, respectively, and $P(e(t) = 1) = p, p \neq 0.5$ for all t. The objective of the attack is to sequentially recover the initial contents (and possibly, the feedback polynomial as well) of each of the component LFSRs from the known key stream (or cipher text) sequence. To identify the actual initial state of each component LFSR, the correlation between the key stream and the LFSR sequence is calculated for each possible LFSR initial state. The correct initial state is assumed to be that for which the correlation the highest.

Such attacks require knowledge of the structure of the key stream generator. If the entire structure of the generator is known and the secret key is only the initial states of the LFSRs then, for a key stream generator consisting of n LFSRs, the total number of keys to be tried in a brute force attack is $\prod(2^{L_i} - 1)$, where L_i is the length of the i^{th} LFSR. Using a divide and conquer attack, where the initial states of each LFSR are determined sequentially, reduces the total number of keys to be searched to $\sum(2^{L_i} - 1)$, a considerable reduction from the brute force attack.

Correlation attack is applied on stream cipher as follows:

- Generate N bits of output stream $\{z_i\}$ for each possible initial states of LFSR
- Set $count = \sum_{j=1,2,\ldots,N} Y^j = \#\left(z^j = a_i^j\right) \quad 1 \le j \le n.$
- Estimate of $X = {count}/{N}$ is used for testing the hypothesis and finding the initial state of LFSR.

For correlation attack to succeed, the suitable length of cipher digits required is

$$N \approx \left[\frac{\frac{1}{\sqrt{2}}\sqrt{ln(R_i 2^{l_i} - 1) + \gamma_0\sqrt{p_e(1-p_e)}}}{p_e - \frac{1}{2}}\right]^2$$

Where $p_e = Prob(z_n \oplus a_n = 0), R_i =$ Feedback connection of i^{th} LFSR, $l_i =$ Length of i^{th} LFSR, $T =$ empirically determined threshold and

$$\gamma_0 = \frac{N(2p_e - 1) - T}{2\sqrt{N}\sqrt{p_e}(1 - p_e)}$$

III. Fast Correlation Attack

One major problem in correlation attacks is that they perform an exhaustive search for an entire part of the initial state, leading to a huge time-complexity. By increasing the length of the LFSR say $l > 40$, it becomes virtually impossible to find the correct initial state by exhaustively searching the entire key space. The fast correlation attacks introduced by Meier and Staffelbach in 1989 considerably reduce the running-time but require a longer segment of known key stream. Meier and Staffelbach presented two algorithms, referred to as A and B, for fast correlation attacks. 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 ($t \le 10$).

The basic idea of Fast Correlation Attack is as follows: Assume N bits of the output sequence z are given, and correlated with probability $p > 0.5$ to an LFSR sequence a. It is assumed that the feedback connection is known. Otherwise also an exhaustive search over all primitive feedback connections is possible as there are only a very limited number of maximum-length feedback connections with few taps. The following principle is used to reconstruct the LFSR sequence a from z in both the algorithms (Algorithms$A$ and B): every digit a_n of a satisfies several parity check equations derived from the feedback polynomial, all of them involving t other digits of a.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014)

We substitute the corresponding digits of z in these equations. Then to check if $z_n = a_n$ we count the number of equations which hold for z_n will agree with a_n.

The Two Phase of Fast Correlation Attack:

The two phases of the fast correlation attack are described below:

Phase I: Finding Parity Check Equations

In the first phase of the algorithm the set of parity check equations is created as follows: Let $g(x) = 1 + c_1x + c_2x^2 + \cdots \ldots \ldots \ldots \ldots \ldots \ldots + c_lx^l$ denote the feedback polynomial of LFSR of length l and t the number of taps of the LFSR i.e. the number of nonzero coefficients of $g(x)$ is $t + 1$, then i^{th} bit of the LFSR sequence, a_i, can be written as

$$a_i = c_1a_{i-1} + c_2a_{i-2} + \cdots \ldots \ldots + c_la_{i-l}$$

Since the weight of $g(x)$ is $t + 1$, we get in this way $t + 1$ different parity check equations for a_i. Secondly, using the fact that $g(x)^j = g(x^j)$ for $j = 2^i$ parity check equations are also generated by repeatedly squaring the polynomial $g(x)$. So if $g_0(x) = g(x)$,we create new polynomials by $g_{k+1}(x) = g_k(x)^2$, $k = 1,2,3, \ldots \ldots \ldots$. This squaring is continued until the degree of a polynomial $g_k(x)$ is greater than the length N of the observed key stream. Each of the polynomials $g_k(x)$ are of weight $t + 1$ and hence each gives $t + 1$ new parity check equations for a fixit position a_n. Combining this squaring technique with shifting we get the set of equations in time, the same parity check equations are essentially valid in each index position of a. The number of parity check equations, denoted m, that can be found in this way is given by:

$$m \approx log_2\left(\frac{N}{2l}\right)(t + 1)$$

Example: For t = 3. Primitive polynomial satisfies recursion: $x_j = x_{j-1} + x_{j-3}(mod\ 2)$. After squaring the recursion $x_j = x_{j-2} + x_{j-6}(mod\ 2)$ does also holds. By shifting, we get

$$x_{j-3} + x_{j-1} + x_j = 0;\ x_{j-2} + x_j + x_{j+1} = 0\ ;x_j + x_{j+2} + x_{j+3} = 0.$$

Phase II: Decoding key stream bits

In the second phase, the m equations for position a_n are written as follows:

$$\begin{aligned} a_n + b_1 &= 0, \\ a_n + b_2 &= 0 \\ \ldots\ldots\ldots\ldots\ldots & \\ \ldots\ldots\ldots\ldots\ldots & \quad (1) \\ a_n + b_m &= 0, \end{aligned}$$

Where each b_i is the sum of t different positions of a, applying the same relations above to the key stream $z_n = z$ (at the same index positions). We get the following expressions

$$z + y_i = L_i\ ,\ i = 1,2, \ldots \ldots .m, \quad (2)$$

Where y_i is the sum of the positions in the key stream corresponding to the positions in b_i and L_i is not necessarily 0. Assume that h out of the m equations in (1) hold, i.e. $h = |\{i: L_i = 0,\ 1 \le i \le m\}|$. Then we calculate the probability p^* for $z_n = a_n$, conditioned on the number of relations satisfied,i.e. $p^* = P(a_n = z_n|h\ \ equation\ holds)$ as

$$p^* = \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)s^{m-h}(1-s)^h}$$

Where $s = s(p,t)$ (the probability of b_i and y_i being equal) can be computed as

$$s(p,t) = ps(p,t-1) + (1-p)\big(1 - s(p,t-1)\big), s(p,1) = p$$

Example: $p = 0.75, t = 2$, LFSR-length $l = 100, N = 5000$ output bits of b. Then $m = 12$, (In the average), and $s = 0.75^2 + 0.25^2 = 0.625$. Value of $p^* = 0.9993$, If $h = 12$ relations.

The following two algorithms were proposed by Meier and Staffelbach for decoding the key stream bits [2, 3].

Algorithm A

In this algorithm we first find the equations to each position of the received bit and evaluate the equations (we can select those bits that satisfy most equations). To obtain an estimate of the sequence a at the corresponding positions we then calculate the probabilities p^* for each bit in the key stream, and select the l positions with highest value of p^* and calculate a candidate initial state. Finally, find the correct value by checking the correlation between the sequence and the key stream for different small modifications of the candidate initial state. This results in a considerably reduced exhaustive search to sort out sufficiently many correct bits, in order to determine the LFSR sequence a by solving linear equations.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014)

Computational Complexity

The computational complexity of the algorithm is of order $O(2^{cl})$, where $c < 1$ is a function of t, p, and N/l.

Example: For $t = 2$ taps, $N/l = 106$, and $p > 0.6$, the number c is smaller than 0.25, and for $p > 0.67$, c is below 0.001. This is a considerable improvement compared with exhaustive search, where $c = 1$. However, for large $t(t > 10)c$ comes very close to $H(p)$, where $H(p)$ denotes the binary entropy function. This proves that Algorithm A for large t gives no advantage over (a modified) exhaustive search.

Algorithm B

In Algorithm B we do not search for the most reliable bits. Instead of calculating the probabilities p^* once and then make a hard decision, the probabilities are calculated iteratively. We calculate the value of m and find the value of $h = h_{max}$ such that $I(p, m, h)$ is maximum. Compute P_{thr} and N_{thr} computed as above. In first round, for every digit of z, we compute the new probability $p^*, (s(p_1 \dots \dots . p_t, t)$ and $s(p_1, 1))$ with respect to the individual number of relations satisfied. Calculate the number of digits N_w is less than the expected number of digits N_{thr} under the same threshold probability then complement those digits of z and reset the probability of each digit to the original value p. If digits of z not satisfying linear relation of LFSR sequence then process is repeated until z satisfies a.

Computational Complexity

The computational complexity of algorithm B is of order $O(l)$ (i.e. linear in length l LFSR, for fixed t, p and N/l. Algorithm B will be able to reproduce the LFSR sequence a within a definite limit which is attained for $t > 10$ if $p < 0.75$.

The algorithms A and B above enable attacks against LFSRs of considerable length (e.g. $l = 1000$ or greater), when the LFSR contains few tapes. They work with LFSRs with many taps as each parity check equation gives a very small average correlation and hence many equations are needed. Algorithm A is preferable if $c \ll 1$ and p is near 0.75, whereas Algorithm B becomes more efficient for probabilities p near 0.5.

IV. Convolutional Codes Based Fast Correlation Attack

Correlation attacks are often viewed as a decoding problem. For an LFSR of length l, the set of possible LFSR sequences is denoted by L, where $|L| = 2^l$. Now for a fixed length N of the ciphertext, the set of all truncated sequences from L is also a linear $[N, l]$ block code, referred to as C. Thus, the LFSR sequence $a = (a_1, a_2, \dots \dots \dots a_N)$ is regarded as a codeword from C and the key stream sequence $z = (z_1, z_2, \dots \dots \dots z_N)$ is regarded as the received channel output. From the definition of the correlation between a_i and z_i, it can describe each z_i as the output from the binary symmetric channel (BSC), when a_i was transmitted. The correlation probability $1 - p$, defined by $1 - p = P(a_i = z_i)$, gives p as the crossover probability (error probability) in the BSC. Without loss of generality it is assumed $p < 0.5$.

The cryptanalyst's problem can be formulated as follows. Given a length N received word $z = (z_1, z_2, \dots \dots \dots z_N)$ as output of the BSC(p), find the length N codeword from C that was transmitted.

Johansson and Jonsson in 1999 introduced the concept of correlation attack using convolutional coding [1, 6]. The parity check equations as described in Fast correlation attack designed for a second phase consists of a very simple memoryless 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.

Phase I: Finding Parity Check Equations to Design Convolutional Encoder

The proposed algorithm transforms a part of the code C stemming from the LFSR sequences into a convolutional code. The encoder of this convolutional code is created by finding suitable parity check equations from C. The convolutional code will have rate $R = 1/(m + 1)$, where the constant $(m + 1)$ will be determined later. Furthermore, let B be a fixed memory size. In a convolutional encoder with memory B the vector v_n of codeword symbols at time n is of the form $v_n = a_n G_0 + a_{n-1} G_1 + \cdots \dots \dots \dots \dots + a_{n-B} G_B$, where in the case $R = 1/(m + 1)$ each G_i is a vector of length$(m + i)$.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014)

The task in the first phase of the algorithm is to find suitable parity check equations that will determine the vectors G_i, $0 \le i \le m$, defining the convolutional code. The steps involved in phase I are as follows:

Step 1: Converting LFSR sequence to linear code

Determine the linear code C stemming from the LFSR. There is a corresponding

$l \times N$ Generator matrix G_{LFSR}. Now $a = a_0 G_{LFSR}$, where a_0 is the initial state of the LFSR. The generator matrix can be written in systematic form, i.e. $G_{LFSR} = (I_l, Z)$ where I_l is the $l \times l$ identity matrix. The parity check matrix is written as $P_{LFSR} = (Z^T, I_{N-1})$.

Example: Consider the primitive polynomial $g(x) = x^4 + x^3 + 1$. The sequence a is generated by the 4×10 matrix G_{LFSR} ,

$$a = a^l G$$

$$= [a_0 \quad a_1 \quad a_2 \quad a_3] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Step 2: Finding parity check equation with in linear block code

All parity check equations in G_{LFSR} can be formed from a current symbol a_n, an arbitrary linear combination of the B previous symbols $a_{n-1}, \dots\dots\dots, a_{n-B}$ together with at most t other symbols. Now generator matrix by indexing $n = B + 1$, where B is parameter chosen by user which will depend on size convolutional decoder, can be written as

$$G_{LFSR} = \begin{bmatrix} I_{B+1} & Z_{B+1} \\ O_{l-B-1} & Z_{l-B-1} \end{bmatrix}$$

Parity check equations for a_{B+1} with weight t outside the first $B + 1$ positions can then be found by finding linear combinations of t columns of Z_{l-B-1} that add to the all zero column vector.For simplicity, consider $t = 2$. In this case, parity check equation is found if two columns from G_{LFSR} have the same value when restricted to the last $l - B - 1$ entries (the Z_{l-B-1} part). Each pair of columns satisfying this equation gives one parity check equation. This gives set of m parity check equations for a_{B+1} and also since LFSR sequences are cyclic (i.e. periodic) therefore exactly the same set of parity checks is satisfied for any index position n which is given by:

$$\begin{gathered} a_n + c_{11}a_{n-1} + c_{21}a_{n-2} \\ + \cdots \dots\dots\dots\dots\dots \\ + c_{B1}a_{n-B} + a_{n+i1} \\ + \quad a_{n+j1} = 0 \\ a_n + c_{12}a_{n-1} + c_{22}a_{n-2} + \cdots \dots\dots\dots\dots + c_{B2}a_{n-B} + a_{n+i2} \\ + a_{n+j2} = 0 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots \qquad (3) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_n + c_{1m}a_{n-1} + c_{2m}a_{n-2} + \cdots \dots\dots\dots\dots + c_{Bm}a_{n-B} \\ + a_{n+im} + a_{n+jm} = 0 \end{gathered}$$

To determine the index positions i and j together with the linear combination of c_i where $1 \le i \le B$ that satisfy above equation, the column vector in G_{LFSR} are numbered as

$$G_{LFSR} = [g_{-B}\ g_{-(B-1)} \dots\dots g_0\ \ g_1 \dots\dots\dots g_{N-(B+1)}]$$

And determine the columns that satisfy the equation

$$\begin{gathered} c_B g_{-B} + c_{(B-1)} g_{-(B-1)} + \cdots \dots\dots + \cdots \dots + c_1 g_{-1} + g_0 \\ + g_i + g_j = 0 \end{gathered}$$

Such that

$$(g_i + g_j)^T = \overbrace{(*,*,*, \dots *,}^{B}\ 1, \overbrace{0,0,0, \dots\dots,0)}^{l-B-1}.$$

After getting the index position i and j determine the column vectors g_{-B} to g_{-1} which will be used in above equation.

Example: Let $t = 2$, and $B = 1$. From Step 1, we see that $(g_5 + g_8)^T = (1,\ 1,\ 1,\ 1) + (1,\ 0,\ 1,\ 1) =$ $(0,\ 1,\ 0,\ 0)$. From this we get $c_0 = 1$, $i_1 = 5$ and $i_2 = 8$ and the equation is $a_0 = a_5 + a_8$. Because of the cycle structure we finally get $a_t = a_{t+5} + a_{t+8}$. This equation will hold for every sequence that is generated with $g(x) = x^4 + x^3 + 1$ as feedback polynomial.

Step 3: Design of convolution Encoder Using Parity Equations

The above parity equations are used to design convolutional encoder $(R = 1/(m+1))$ with the following generator matrix

$$\begin{bmatrix} G_0 \\ G_1 \\ . \\ . \\ G_B \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots\dots\dots & 1 \\ 0 & c_{11} & c_{12} & \dots\ \dots & c_{1m} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & c_{B1} & c_{B2} & \dots\dots\dots & c_{Bm} \end{bmatrix}$$

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014)

Step 4: Transform the cipher text z_i to the received stream r_i

For each defined codeword symbol v_n^i in the convolutional code one has an estimate of that symbol from the transmitted sequence z. We refer to this sequence as the received sequence and denote it by$(r_1, r_2, \dots\dots, r_N)$, where $r_i = (r_i^0, r_i^1, \dots\dots\dots, r_i^m)$.From the right hand side of (3), we have $v_n^k = \sum_{j=1}^{2} a_{n-B-1+i_j^k}$, $1 \le k \le m$. The received sequence r_i can be constructed from the ciphertext z_i is (by replacing $a = z$ in the above equation i.e. $r_n^k = \sum_{j=1}^{2} z_{n-B-1+i_j^k}, 1 \le k \le m$, $r_n^0 = z_n$, $r_n^1 = z_{n+i1} + z_{n+j1}$,.....

....., $r_n^m = z_{n+im} + z_{n+jm}$.

The index positions $(i_1, i_2, \dots\dots\dots, i_m)$ and $(j_1, j_2, \dots\dots\dots, j_m)$ are same as specified in equation (3). The above stream r is now used to construct the received stream z with a length of at least $(m+1).l$ from the cipher text.

Phase II: Decoding with Viterbi Algorithm

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. For each possible starting state $B = (s_1, s_2, \dots\dots\dots, s_B)$, let $d_H(B, z_B)$, where $z_B = (z_1, z_2, \dots\dots, z_B)$ and $d_H(B, z_B)$ denotes the Hamming distance between B and z_B be initial metric for that state when we start the Viterbi algorithm at $n = B$. Then one runs the Viterbi algorithm over l information symbols. At depth $B+1$ we search for the ending state $B+1$ with minimum metric. The decoder output is then the information sequence corresponding to the surviving path from one of the starting states B to the ending state $B+1$ 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 $log\left(P\left(s = (z_1, z_2, \dots.., z_B)\right)\right)$ be the initial metric for that when we start the Viterbi algorithm at $n = B$.
2. Decode the received sequence r using the Viterbi algorithm from $n = B$ until $n = J$.

Output the estimated information sequence $(\hat{a}_{B+1}, \hat{a}_{B+2}, \dots\dots\dots, \hat{a}_{B+l})$.

Finally, calculate the corresponding initial state of the LFSR.

V. Conclusion

The correlation attack by Seigenthaler and Fast correlation attack by Meier and Stafflebach were briefly presented. In was shown that correlation attack can be viewed as the problem of decoding a given vector to codeword from linear codes. The Fast correlation attack works well when the polynomial that defines the LFSR has few taps, but fails when the polynomial has many taps. Thomas Johansson and Fredrik Jonsson's attack works for LFSRs with many taps. An efficient search algorithm is used to find parity equations that are suitable for convolutional codes. The decoding is done using the Viterbi algorithm, which is maximum likelihood.

REFERENCES

[1] C. S. Bruwer, "Correlation attack on stream ciphers using convolution codes" M. E. Dissertation.

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

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

[4] T. Siegenthaler, "Correlation-immunity of non linear combining functions for cryptographic applications", IEEE Trans. On Information Theory, vol. It-30, 1984, pp. 776-780.

[5] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only", IEEE Trans. On Computers, vol. C-34, 1985, pp. 81-85.

[6] T. Johansson and F. Jonsson. "Improved fast correlation attacks on stream ciphers via convolutional codes". In Advances in Cryptology- EUROCRYPT-99, pages .301-314. Springer-Verlag, 1999.