A Study on Parity Checks in Stream Cipher Correlation Attacks

Jun-Chu Hong , Wen-Nung Tsai , Rong-Jaye Chen

Department of Computer Science and Information Engineering,
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
Email: {hongjc, tsaiwn, rjchen} @csie.nctu.edu.tw

ABSTRACT

The commonest stream cipher system uses a keystream generator which consists of several LFSRs combined by a combining function. If there exists a measure of correlation between the output sequence of the keystream generator and an arbitrary LFSR, the initial state of the LFSR can be reconstructed by a correlation attack, that is, the partial key in the LFSR is determined. W. Meier and O. Staffelbach proposed a correlation attack method using parity check equations.

In this paper, we discuss the algorithm and its constraints, and then propose some improvements: computing more low-weight parity check equations, accounting the precise number of relations of each digit, and solving the system of linear independent equations from digits instead of calculating the whole output sequence and the initial state of the LFSR from the relations among the digits.

1.Introduction

In cryptography, there are two basic types of symmetric encryption/decryption algorithms: block ciphers and stream ciphers [14][15]. *Block ciphers* operate on blocks of plaintext and ciphertext. The same plaintext block will be always encrypted to the same ciphertext block, using the same key. *Stream ciphers* operate on streams of plaintext and ciphertext one bit or byte at a time. The same plaintext bit or byte will be encrypted to a different bit or byte every time it is encrypted. That is, for a plaintext string $s = s_1 s_2 \ldots s_L$ in a block cipher system, the ciphertext string c is obtained as follows.

$$c = c_1 c_2 \ldots c_L = E_K(s_1)\ E_K(s_2) \ldots E_K(s_L)$$

And for a plaintext string $s = s_1 s_2 \ldots$ in a stream cipher system, the ciphertext string c is obtained as follows.

$$c = c_1 c_2 \ldots = E_{z_1}(s_1)\ E_{z_2}(s_2) \ldots$$

where the keystream (or called 'running key') z^{∞} is $z_1 z_2 \ldots$, the actual key is K, and the state of the encryption device is σ_j which may be dependent on K, $\sigma_{j\text{-}1}$ and $s_{j\text{-}1}$, and the function f_j is used to generate z_j (the j^{th} element of the keystream) such that

$$z_j = f_j(K, \sigma_j)$$

Thus a stream cipher system keeps a state σ_j in the memory, when a block cipher system does not. The essential difference between block and stream ciphers is the usage of memory as is shown in Figure 1.1.

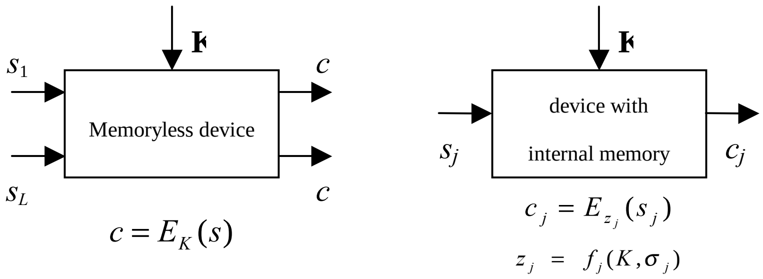

Figure 1.1. The difference between block and stream ciphers

Obviously, we can think of a block cipher as a special case of a stream cipher where the keystream is constant: $z_j = K$. In this paper, we focus on stream cipher systems and correlation attacks on stream ciphers.

There are two different approaches to stream encryption: synchronous methods and self-synchronous methods [16][17]. In a *synchronous stream cipher*, as shown in Figure 1.2, the next state depends only on the previous state and not on the input so that the succession of states is independent of the message stream. The keystream is therefore generated independently of the message stream. Thus, if a ciphertext character is lost during transmission, the sender and receiver must resynchronize their generators before they proceed further.

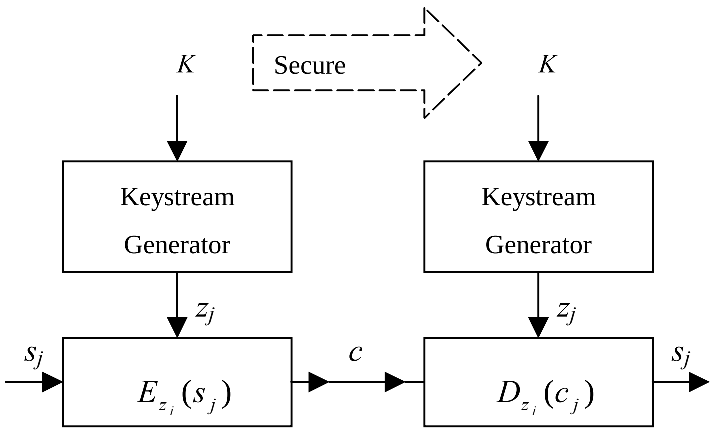

Figure 1.2 Synchronous stream ciphers

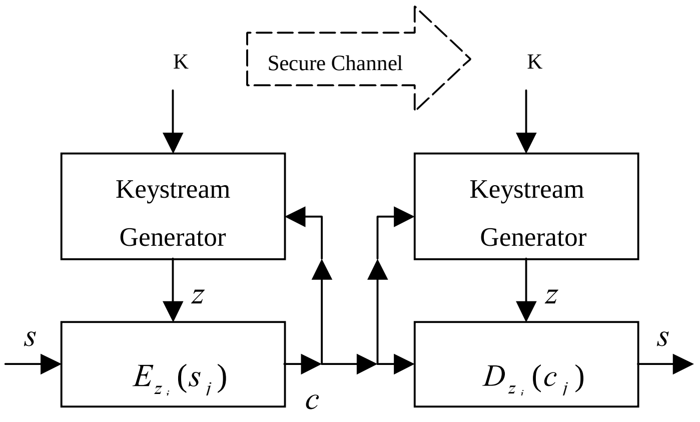

Figure 1.3. Self-synchronous stream ciphers

In a *self-synchronous stream cipher*, as shown in Figure 1.3, each keystream character is derived from a number *n* of preceding cipher characters. Thus, if a ciphertext character is lost or modified during the transmission, the error propagates forward for *n* characters, but the cipher resynchronizes itself after *n* correct ciphertext characters have been received.

The algorithm that generates the keystream must be deterministic so that the stream can be reproduced for decipherment. One important kind of synchronous stream cipher is the *additive synchronous stream ciphers*, where the characters of the keystream are from an Abelian group (G,+) and the ciphertext character c_j is the addition of the keystream character z_j and plaintext stream character s_j ($c_j = s_j + z_j$, $s_j = c_j - z_j$ and "-" means the inverse operation of "+") . In this thesis, we only discuss GF(2) , so the effects of "+" and "-" are both the same with XOR ($c_j = s_j + z_j = s_j - z_j = s_j \oplus z_j$).

Finite state machines are important mathematical objects for modeling electronic hardware. Furthermore, due to their recursive feature, finite state machines are convenient means for realizing infinite word-functions built over finite alphabets. Many keystream generators can be modeled by finite state machines. In a synchronous stream cipher, the keystream generator may be viewed as an autonomous finite state machine as depicted in Figure 1.4.

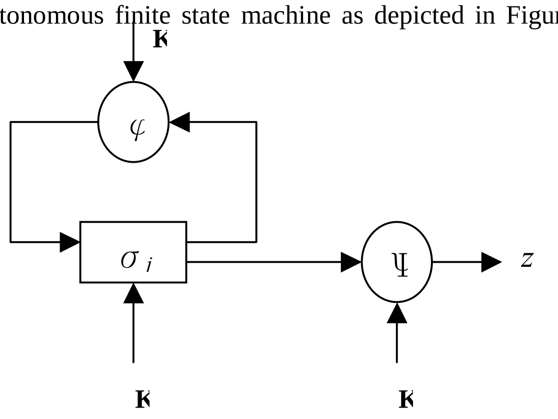

Figure 1.4. Keystream generators as autonomous finite state machines

The keystream generator as a finite state machine consists of an output alphabet $\{z_j\}$ and a state set $\{\sigma_j\}$, together with two functions (φ, Ψ) and an initial state σ_0. The *next state function* φ maps the current state σ_j into a new state σ_{j+1} from the state set. The *output function* Ψ maps the current state σ_j into an output symbol z_j from the output alphabet. The key *K* may determine the next state function φ and the output function Ψ as well as the initial state σ_0.

The major purpose of designing a keystream generator is to prevent from an attacker to predict the output sequence *z*. So the output sequence of the keystream generator should satisfy some cryptographic requirements such as long period, large linear complexity, good auto correlation, uniform pattern distribution (randomness) , and so on.

2.Linear Feedback Shift Register based Stream Ciphers

Linear Feedback Shift Registers (LFSRs) are the commonest components in stream cipher systems since they can generate binary sequences speedily. Figure 2.1 is the structure of a LFSR [16][17].

A linear feedback shift register of *length L* consists of *L* stages $S_1 \sim S_L$. Each stage stores one bit. During each unit of time, the following operations are executed :

(1) The content of S_1 is output and forms the output sequence of the LFSR.

(2) The content of S_i is shifted to S_{i-1} , for $2 \leqq i \leqq L$.

(3) The new content of S_L is calculated by $\sum_{i=1}^{L} -C_i\, S_{L+1-i}$

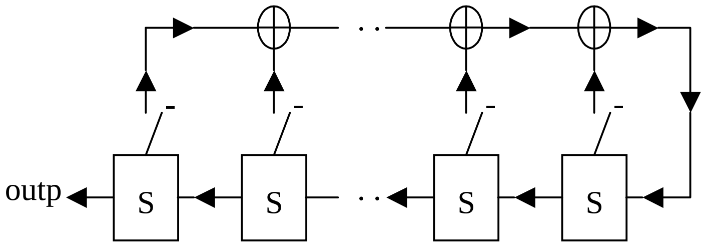

Figure 2.1. The Structure of a Linear Feedback Shift Register

So, the j^{th} digit (bit) s_j ($j > L$) of the output sequence *s* of the LFSR can be calculated from $s_j = \sum_{i=1}^{L} -C_i\, s_{j-i}$. In GF(2), $s_j = \sum_{i=1}^{L} -C_i\, s_{j-i} = \sum_{i=1}^{L} C_i\, s_{j-i}$. We use a polynomial $c(x) = C_L x^L + C_{L-1} x^{L-1} + \ldots + C_1 x + 1$ to record the structure of the LFSR.

Definition. The initial content of the LFSR is called the *initial state* of the LFSR. In general, the initial state of the LFSR is the key or a part of the key of the stream cipher system.

Definition. The polynomial $c(x) = C_L x^L + C_{L-1} x^{L-1} + ... + C_1 x + 1$ is called the *connection polynomial* of the LFSR. If the degree of $c(x)$ is L, that is $C_L = 1$, then the LFSR is *nonsingular* and the output sequence of the LFSR is periodic.

Example. Consider the LFSR in Figure 2.2.

The initial state of the LFSR of length 4 is 0, 1, 1, 0. The connection polynomial is $c(x) = x^4 + x + 1$. And for $j > 4$, the jth output digit of the LFSR is $s_j = s_{j-1} + s_{j-4}$. The output sequence is s = 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, …, and is periodic with period 15 .

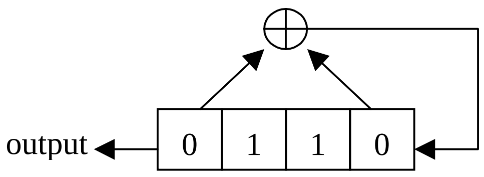

Figure 2.2. An illustration of a LFSR

Theorem. The period of a sequence generated by a non-singular LFSR of length L is at most 2^L-1 .

Since a LFSR of *length L* consists of L stages, the number of the contents of the LFSR is 2^L. The next state of L-zeros is still all-zeros. So, a LFSR with all-zeros as its initial state can only generate a sequence of zeros, and the period is 1. If the initial state of a LFSR is not all-zeros, there are at most 2^L-1 possible contents of the LFSR and the period of the change of the LFSR's content is at most 2^L-1. The output digit of a LFSR each step is dependent only on the previous state of the LFSR. So, the period of the output sequence of a non-singular LFSR of length L is at most 2^L-1, too.

A sequence generated by a non-singular LFSR of length L is called a *maximal sequence* or *m-sequence* if its period is 2^L-1, and the LFSR is called a *maximal-length LFSR*. Every m-sequence satisfies Golomb's randomness postulates and is also a pn-sequence.

But, how to find a maximal-length LFSR ?

Definition. A polynomial of degree n is called *irreducible* if it cannot be factored.

Definition. An irreducible polynomial of order n is called *primitive* if and only if it divides x^p+1 for only a p which is greater than or equal to 2^n-1 .

In order to examine whether an irreducible polynomial $f(x)$ of degree n in GF(2) is primitive, one can compute $g_i(x) = x^{\frac{2^n-1}{r_i}}$ mod $f(x)$ for all distinct prime factors $r_1, r_2, ..., r_t$ of 2^m-1 . If any $g_i(x)$ = 1, then $f(x)$ is not a primitive polynomial, else it is primitive.

Fact. If the connection polynomial of a LFSR of length L is a primitive polynomial of degree L, then each of the 2^L-1 non-zero initial states of the non-singular LFSR generates an output sequence with period 2^L-1 .

Since every m-sequence satisfies Golomb's randomness postulates, it seems that we can take a LFSR with a primitive connection polynomial as a keystream generator. But it is not secure enough. In 1976, Abraham Lempel and Jacob Ziv [4] proposed to use the linear complexity of the keystream as a measure of the strength of a stream cipher system.

Definition. The *linear complexity* of a finite binary sequence s of length n, denoted as $\Lambda(s)$, is the length of the shortest LFSR that can generate a sequence having s as its first n digits. And $\Lambda(s) = 0$ if s is an empty string.

Fact. Suppose that the linear complexity of a binary sequence s is $\Lambda(s)$. As long as we observe consecutive 2 $\Lambda(s)$ digits of a subsequence of s, we can calculate $\Lambda(s)$ and the shortest LFSR which can generate s. This means that although the period of the output sequence s of a maximum-length LFSR of length L reaches to 2^L-1, the whole sequence will be disclosed if any subsequence of length $2L$ of s is known.

Berlekamp and Massey [1] proposed an efficient algorithm to determine the linear complexity of a finite binary sequence s of length N.

From the discussions above, we know that the keystream generated by a maximum-length LFSR is still not secure enough. One technique destroying the linearity inherent in LFSRs is to generate the keystream by some nonlinear function of the stages of a LFSR. Figure 2.3 shows the structure. The kind of keystream generator is called a *nonlinear filter generator* and the function f is called the *filter function*.

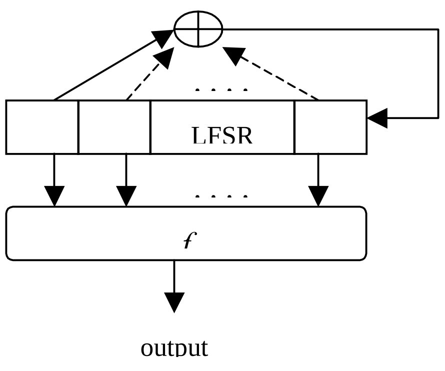

Figure 2.3. A filter generator

Fact. The linear complexity of the keystream generated by a nonlinear filter generator with a LFSR of length *L* and a filtering function *f* of nonlinear order *m* is at most $\sum_{i=1}^{m}\binom{L}{i}$.

Adding a filter function to a LFSR may increase the linear complexity of the output sequence, but the period is at most still the same.

Another general technique for destroying the linearity inherent in LFSRs is to generate the keystream by a nonlinear function *F* of the outputs of several LFSRs. Figure 2.4 shows the structure. The kind of keystream generator is called a *nonlinear combination generator* and the function *F* is called the *combining function*.

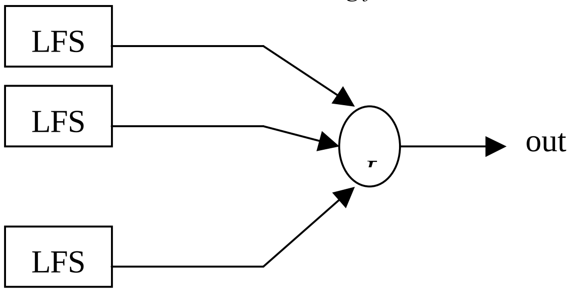

Figure 2.4. A combination generator

Every Boolean function $F(x_1,x_2,\ldots,x_m)$ can be written as a modulo 2 sum of distinct m^{th} order products. The expression is called the *algebraic normal form* of *F*.

Fact. Suppose that *m* LFSRs, whose lengths $L_1, L_2, \ldots, L_m$ are pairwise distinct and greater than 2, are combined by a nonlinear function $F(x_1,x_2,\ldots,x_m)$ which is expressed in algebraic normal form. The linear complexity of the keystream is $F(L_1,L_2,\ldots,L_m)$.

3. Correlation Attacks

In conventional cryptography, pseudo-noise generators (pn-generators) consist of *m* linear feedback shift registers of length L_i ($i = 1, 2, \ldots, m$) and a known combining function *F* (see Figure 2.4). To avoid a cryptanalytic attack using Berlekamp-Massey algorithm, the combining function *F* should be nonlinear.

The initial state and feedback connection polynomial of LFSR$_i$ are referred to as the LFSR$_i$ part of the key. Assume that the feedback connection polynomials of all LFSR's of length L_i ($i = 1, 2, \ldots, m$) are primitive. So LFSR$_i$ of length L_i has $2^{L_i}-1$ different possible initial states and the number R_i of different primitive feedback connection polynomials for an LFSR equals $\frac{\varphi(2^{L_i}-1)}{L_i}$. Hence, there are $R_i(2^{L_i}-1)$ possibilities for the LFSR$_i$ part of the key and the total number *K* of the keys for the pseudo-noise generator in Figure 2.4 is

$$K=\prod_{i=1}^{m}R_i(2^{L_i}-1).$$

In a brute force attack and a worst case situation, all of the *K* keys have to be examined, which is not feasible.

However, there may be correlation between some of the inputs S^i and the output *Z*. T. Siegenthaler [7] proposed a divide and conquer correlation attack method that the LFSR$_i$ part of the key would be found independently from the LFSR$_j$ parts ($j=1, \ldots, m$; $j \neq i$) with approximately $R_i(2^{L_i}-1)$ tests. So, the number of trials can be reduced from $\prod_{i=1}^{m}R_i(2^{L_i}-1)$ to approximately $\sum_{i=1}^{m}R_i(2^{L_i}-1)$.

A few correlation attack methods were proposed after T. Siegenthaler showed that it is possible to independently reconstruct the initial state of each LFSR combined by a combining function with the divide and conquer correlation attack method if there exists a measure of correlation between the keystream sequence and the outputs of the LFSRs. In 1988, a fast correlation attack using parity check equations was proposed by Willi Meier and Othmar Staffelbach [8][9].

The model is also the commonest type of keystream generators that consist of *m* LFSRs whose output sequences are combined by a nonlinear function *F*.

Let the correlation probability between the output sequence *z* of the keystream generator and the output sequence *a* of a LFSR be larger than 0.5 . Suppose that *N* digits of the output sequence *z* of the combining function are given, the feedback connection polynomial of the LFSR with *t* taps and length *L* is known, and the LFSR generates a sequence *a*.

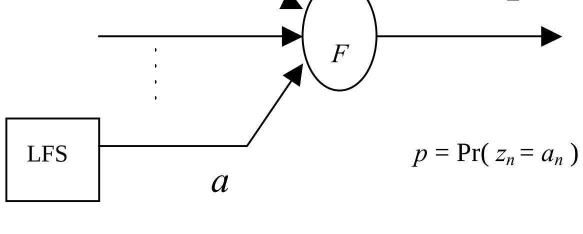

Figure 3.1. A model of Meier-Staffelbach algorithm

By iterated squaring of the feedback connection polynomial of the LFSR, an amount of linear relations will be generated for every digit a_n, and each relation contains

t+1 digits of the sequence a. For example, the feedback connection polynomial $c(x)$ with 2 taps of a LFSR of length 4 is x^4+x^3+1, then we know a linear relation $a_n = a_{n-3}+a_{n-4}$. By shifting the index, we can get the other two linear relations $a_{n+3}=a_n+a_{n-1}$ and $a_{n+4}=a_{n+1}+a_n$ that contain a_n. Using the fact that in GF(2), $c(x)^j = c(x^j)$ for $j = 2^i$, we can get more parity check equations i.e. $(x^4+x^3+1)^2 = x^8+x^6+1$ is also a polynomial of weight t+1. Therefore, we can generate t+1 parity check equations for a fixed position a_n.

The average number m of the relations can be computed as

$$m = m(N,L,t) = (t+1)\cdot\log_2\left(\frac{N}{2L}\right) .$$

Thus for a fixed position a_n, we can write the m relations as

$$a_n + b_1 = 0 \text{ , } a_n + b_2 = 0 \text{ , } . . . \text{ , } a_n + b_m = 0$$

where each b_i is the sum of t other different positions of the sequence a.

Applying the same relations to the corresponding positions of the keystream z, we also get m relations as

$$z_n + y_1 = L_1 \text{ , } z_n + y_2 = L_2 \text{ , } . . . \text{ , } z_n + y_m = L_m$$

where each y_i is the sum of t other different positions of the sequence z.

The idea of Meier and Staffelbach is that the more parity check equations of z_n held, the higher is the probability of $z_n = a_n$. Then we use the digits of the sequence z which are likely to be same with the corresponding digits of the sequence a to reconstruct the sequence a. And the initial state of the LFSR is figured out.

We know that the probability p = Pr($z_n = a_n$) and s = Pr($y_i = b_i$) = $s(p,t)$ can be computed from the recursion: $s(p,t) = p\ s(p,t\text{-}1) + (1\text{-}p)(1\text{-}s(p,t\text{-}1))$, $s(p,1) = p$, we can calculate these probability variables :

p^* = Pr($z_n = a_n$ | exactly h out of the m relations are satisfied $L_i = 0$)

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

Q(p,m,h) = Pr(at least h out of the m relations are satisfied $L_i = 0$)

$$= \sum_{i=h}^{m} \binom{m}{i} \left(p\ s^i(1-s)^{m-i} + (1-p)(1-s)^i s^{m-i} \right)$$

R(p,m,h) = Pr($z_n = a_n$ ∩ at least h out of the m relations are satisfied $L_i = 0$)

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

T(p,m,h) = Pr($z_n = a_n$ | at least h out of the m relations are satisfied $L_i = 0$)

= R(p,m,h) / Q(p,m,h)

Since $p > 0.5$ and t is even, as h grows, p^* grows, that is, the probability of $z_n = a_n$ increases. Thus the idea of Meier and Staffelbach is proved valid. We want to choose enough digits of sequence z to reconstruct the sequence a, hence we have to determine the maximum value of h such that $Q(p,m,h)\cdot N \geqq L$. After deciding the value of h, we search for the digits of the sequence z which satisfy at least h parity check equations as a reference guess I_0 of the sequence a at the corresponding positions to reconstruct the sequence a by the relations. We can also estimate the average number r of errors of I_0 by $r = (1 - T(p,m,h)) \bullet L$. If $r << 1$, these digits are likely correct. We can examine whether the initial state we calculate from I_0 is correct by computing correlation and comparing it with the threshold which we describe in the previous section. If it is wrong, find the correct guess by testing modifications of I_0 which has Hamming distance 1, 2, … until a correct one is obtained.

4. Improve the Meier-Staffelbach Algorithm

Although Meier-Staffelbach algorithm is efficient, there are still some defects. We will discuss these defects and propose improvements to lower the influence caused by these defects.

Defect 1. The number t of the taps should be small.

First, let's consider the probability Pr($y_i = b_i$). It can be computed from the recursion: Pr($y_i = b_i$) = s = $s(p,t) = p\ s(p,t\text{-}1) + (1\text{-}p)(1\text{-}s(p,t\text{-}1))$ and $s(p,1) = p$. By the recurrence relation, we can solve s:

$$s(p,t)-(2p-1)\ s(p,t-1) = 1-p$$

$$(2p-1)\ [\ s(p,t-1)-(2p-1)\ s(p,t-2)\ ] = (2p -$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$(2p-1)^{t-2}\ [\ s(p,2)-(2p-1)\ s(p,1)\ ] = (2p$$

$$s(p,t)-(2p-1)^{t-1}\ s(p,1) = (1-p)\ [1+(2p-1)+...+$$

$$= (1-p)\ \frac{1-(2p-1)^{t-1}}{1-(2p-1)} = \frac{1-(2p-1)^{t-1}}{2}$$

$$\Rightarrow s(p,t) = (2p-1)^{t-1}\cdot p + \frac{1-(2p-1)^{t-1}}{2} = \frac{1+(2p-1)^{t}}{2}$$

So, if t is too large, the probability Pr($y_i = b_i$) = $s(p,t)$ will approximate to 0.5 and $p^* = \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h}+(1-p)(1-s)^h s^{m-h}} \approx \frac{p(0.5)^h(0.5)^{m-h}}{p(0.5)^h(0.5)^{m-h}+(1-p)(0.5)^h(0.5)^{m-h}} = p$.

This means that no matter how many parity check equations of a digit are held, the probability p^* approximates to a constant p = Pr($z_n = a_n$) if t is too large. Therefore the probability p^* doesn't give us more information than the probability Pr($z_n = a_n$) such that Meier-Staffelbach algorithm becomes an exhaustive search.

Besides, when we want to determine the value of a certain digit, we need to find a parity check equation which contains the digit and another t digits believed to be correct. So it is infeasible if t is large.

Defect 2. $\frac{N}{L}$ should be neither too small nor too large.

From the formula $m = (t+1)\cdot\log_2(\frac{N}{2L})$, we know that if $\frac{N}{L}$ is too small, the number of parity check equations we can get by iterated squaring of the connection polynomial would be small such that the accuracy probability of the digits which have the most parity check equations held would be low.

If N is too large, the number Q(p,m,h) · $N \approx L$ of the digits that we believe to be correct is small relatively. There may be not enough relations among these reference digits to determine the whole sequence a and the initial state of the LFSR.

Meier-Staffelbach algorithm requires the number t of taps of the connection polynomial to be small – typically less than 10 . Even t is small, there may exist some parity check equations with weight t+1 or less than t+1, but we can't get them by iterated squaring of the connection polynomial. For example, suppose that the connection polynomial of a LFSR is $c(x) = x^5 + x^4 + x^3 + x^2 + 1$. It is primitive and the period of the output sequence of the LFSR is 2^5 - 1 = 31 . By iterated squaring of the connection polynomial, we have only three parity check equations: $c(x) = x^5 + x^4 + x^3 + x^2 + 1$, $(c(x))^2 = x^{10} + x^8 + x^6 + x^4 + 1$ and $(c(x))^4 = x^{20} + x^{16} + x^{12} + x^8 + 1$, that mean $a_n + a_{n+1} + a_{n+2} + a_{n+3} + a_{n+5} = 0$, $a_n + a_{n+2} + a_{n+4} + a_{n+6} + a_{n+10} = 0$ and $a_n + a_{n+4} + a_{n+8} + a_{n+12} + a_{n+20} = 0$ respectively. But there are still many weight-5 parity check equations like $a_n + a_{n+1} + a_{n+5} + a_{n+6} + a_{n+9} = 0$, … etc. And there exists even weight-3 parity check equations like $a_n + a_{n+1} + a_{n+12} = 0$, $a_n + a_{n+3} + a_{n+8} = 0$, $a_n + a_{n+5} + a_{n+28} = 0$, and etc.

Hence W. T. Penzhorn [11][12] proposed a long division algorithm to compute low-weight parity check equations.

Suppose that the connection polynomial of the LFSR of length L is $c(x)$ and the number of digits of the LFSR's output sequence a we observe is N, and $N \leqq 2^L$-1 . We want to compute some parity check equations of weight w.

For example, given the connection polynomial $c(x) = x^4 + x + 1$, we want to compute weight-3 parity check equations, and choose $\nu_{\max}$ = 14 . When j = 13, we obtain a remainder x^2 which is a single term. So $x^{14} + x^{13} + x^2$ is a parity check equation and $a_n + a_{n+1} + a_{n+12} = 0$. When j = 12, we also obtain a parity check equation $x^{14} + x^{12} + x$ which means $a_n + a_{n+2} + a_{n+9} = 0$.

If w > 3, we may obtain more than one parity check equation in the Step 3 each round. In the above example, if we want to compute weight-4 parity check equations, and choose $\nu_{\max}$ = 14, we obtain $x^{14} + x^{13} + x^{11} + x^9$ and $x^{14} + x^{13} + x^6 + x^5$ when j = 13 .

Compute Weight-3 Parity Check Equations

Since there exists at most one i such that $x^{\nu_{max}} + x^j + x^i$ is a parity check equation, if we want to compute weight-3 parity check equations, Step 3 of Penzhorn's long division algorithm can be modified as follows.

Step 3. Use long division to divide $x^{\nu_{max}} + x^j$ by $c(x)$ until a single-term remainder x^i is obtained. And $x^{\nu_{max}} + x^j + x^i$ is a parity check equation.

However Penzhorn's long division algorithm is not efficient to compute weight-3 parity check equations. If we make some preparations, we can speedily determine the value of i if there exists such an $i \leqq N$.

Each digit a_n can be expressed as a linear combination of the initial state $S_1 \sim S_L$: $a_n = e_{1,n}S_1 + e_{2,n}S_2 + \ldots + e_{L,n}S_L$. And $e_{i,n}$ is the n^{th} output digit of the LFSR with the initial state of all 0's except $S_i = 1$. Therefore, the coefficients $e_{1,n}$, $e_{2,n}$, ..., $e_{L,n}$ can be determined easily. If $x^{\nu_{max}} + x^j + x^i$ is a parity check equation, then $a_n + a_{n+\nu_{max}-j} + a_{n+\nu_{max}-i} = 0$ and $e_{d,n} + e_{d,n+\nu_{max}-j} + e_{d,n+\nu_{max}-i} = 0$ for $1 \leqq d \leqq N$. As long as we record and sort $< e_{1,n}$, $e_{2,n}$, ..., $e_{L,n} >$ for all $n \leqq N$ in preparation, when given a set of $< e_{1,i}$, $e_{2,i}$, ..., $e_{L,i} >$, we can speedily search for the value of i or determine whether there exists such $i \leqq N$ using binary search in time complexity O(log$N$).

The formula $m = (t+1)\cdot\log_2(\frac{N}{2L})$ is to calculate the 'average' number of the relations of each digit. Many digits have more than $(t+1)\cdot\log_2(\frac{N}{2L})$ relations. The maximum value of h which satisfies Q(p,m,h) · $N \geqq L$ is usually too small such that the number of the reference digits which have at least h parity check equations held is much more than L. Therefore, in practice we should not adopt the value of h from the above formula, and we distinctly count the number of held parity check equations of each digit, and then determine the maximum value of h such that the number of digits which have at least h parity check equations held is equal to or greater than L.

Let's look at the result of a simulation program to compare the modified version with the original Meier-Staffelbach algorithm. We set the probability Pr($z_n = a_n$) = 0.75 and the connection polynomial $c(x) = x^{100} + x^{37} + 1$ which is primitive, and decide to observe 20,000 digits. The program randomly chooses an initial state, calculates the 20,000-digit output a of the LFSR and the combining function's 20,000-digit output sequence s which is correlated to a with a probability 0.75, and then counts and records the number of parity check equations held of each digit. We run the program 10,000 times and we found we got better results. Due to the limited space the results are not shown here.

In the original Meier-Staffelbach algorithm, according to the formula $m = (t+1)\cdot\log_2(\frac{N}{2L})$, the maximum value of h which satisfies Q(p,m,h) · $N \geqq L$ is 16 .

In our simulation result, the original Meier-Staffelbach algorithm chooses all correct reference digits in only 917 times of 10,000 simulations because the value of h we determine is too small such that we choose too many digits as the reference digits.

The modified Meier-Staffelbach algorithm chooses all correct reference digits in 9,798 times of 10,000 simulations.

The modified Meier-Staffelbach algorithm we proposed chooses fewer digits as the reference digits to prevent from error digits. But it isn't always successful to calculate the initial state from the relations among the reference digits.

For example, suppose that the connection polynomial is $c(x) = x^7 + x + 1$. We observe 127 digits of the output sequence z of the combining function: 0111110101100010001110111000000001111111000100101 $010011\underline{\underline{0}}01000000\,1\underline{\underline{1}}10\underline{\underline{0}}0110111\underline{\underline{1}}10011101000110110001$ $0011\underline{\underline{1}}100\underline{\underline{0}}\underline{\underline{0}}1011000\underline{\underline{1}}10000010100110$.

By iterated squaring of the connection polynomial, we know these parity check equations: $a_n + a_{n+6} + a_{n+7} = 0$, $a_n + a_{n+12} + a_{n+14} = 0$, $a_n + a_{n+24} + a_{n+28} = 0$, $a_n + a_{n+48} + a_{n+56} = 0$ and $a_n + a_{n+96} + a_{n+112} = 0$. Only z_{104} has 11 relations held. No digit has more than 11 relations held. And z_{55} , z_{65} , z_{68} , z_{76} , z_{100} , z_{105} and z_{113} have 10 relations held respectively. Then we choose these digits and determine the corresponding positions (a_{55} , a_{65} , a_{68} , a_{76} , a_{100} , a_{105} and a_{113}) of the output sequence a of the LFSR as a reference. Although we choose more than 7 digits, only a_{20} , a_{57} , a_{98} , a_{121} , a_{90} , a_{43} and a_{86} can be calculated from the relations of the reference digits. Therefore, in this case, the initial state cannot be determined by the modified Meier-Staffelbach algorithm.

As we discuss before, each digit a_n can be expressed as a linear combination of the initial state $S_1 \sim S_L$: $a_n = e_{1,n}S_1 + e_{2,n}S_2 + \ldots + e_{L,n}S_L$. It is known that a system of L independent linear equations in L unknowns can be solved. Hence if we select L independent digits of the sequence a, we are able to solve $S_1 \sim S_L$ directly by Gaussian elimination rather than calculating the whole sequence a to determine $S_1 \sim S_L$ by the relations. On the contrary, if less than L independent digits are selected, we cannot determine $S_1 \sim S_L$ exactly and have to guess some of them. And if we select some dependent but wrong digits, the system of linear equations may have no solution. With the concept, we know that the key point of choosing digits is not to enlarge the number of chosen digits but to select enough (at least L) independent digits. And we should be able to calculate the value of a digit from the digits those are dependent with it. Hence, superfluous dependent digits are not only useless for solving the system of linear equations but may also hamper the system from having solution.

Our improved algorithm is then as follows.

Step 1. Calculate the number of parity check equations held of each digit.

Step 2. Determine the coefficients $e_{1,n}$, $e_{2,n}$, …, $e_{L,n}$ of each digit a_n's linear combination of the initial state $S_1 \sim S_L$.

Step 3. According to the number of parity check equations held of each digit, we select L independent digits of the sequence z with relations held decreasingly. Use these digits as a reference guess I_0 of the sequence a at the corresponding positions. Then solve the system of L independent linear equations to determine $S_1 \sim S_L$.

Step 4. Find the correct guess by examining modifications of I_0 having Hamming distance 0, 1, 2, …, by the correlation between the sequence and the output sequence of the LFSR with the initial state we calculated.

5. Conclusions

In this paper, we discuss the requirements of a keystream generator in a stream cipher system and the structure and properties of an LFSR. The output sequence of an LFSR has a large period and is unpredictable. But it is easy to attack a LFSR using Berlekamp-Massey algorithm. So most stream cipher systems adopt the nonlinear combination generators that consist of several LFSRs. Correlation attacks are the most popular methods used to attack a stream cipher system. Willi Meier and Othmar Staffelbach proposed a fast correlation attack method using parity check equations. An important design criterion requires that there should be very low correlation between the keystream of the combining function and the output sequence of an arbitrary LFSR of short length otherwise Meier-Staffelbach algorithm could be used to attack the stream cipher system.

We show that it is easy to find low-weight parity check equations no matter how many taps of the LFSR are. In particular, all weight-3 parity check equations, if exist, can be obtained. We also adopt another strategy to choose digits as a reference. Viewing each output digit of the LFSR as a linear combination of the initial state, we choose exactly L independent and most likely correct digits, and solve the system of linear equations to determine the initial state rather than calculating the whole output sequence and the initial state of the LFSR from the relations among the reference digits. Therefore, our algorithm can avoid the situation where the reference digits are not enough to calculate the initial state in Meier-Staffelbach algorithm.

REFERENCES

[1] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, Vol. IT-15, 1969, pp. 122-127.

[2] Edward J. Groth, "Generation of binary sequences with controllable complexity," *IEEE Transactions on Information Theory*, Vol. IT-17, No. 3, May. 1971, pp. 288-296.

[3] Abraham Lempel, "Analysis and synthesis of polynomials and sequences over GF(2)," *IEEE Transactions on Information Theory*, Vol. IT-17, No. 3, May. 1971, pp. 297-303.

[4] Abraham Lempel, and Jacob Ziv, "On the complexity of finite sequences," *IEEE Transactions on Information Theory*, Vol. IT-22, Jan. 1976, pp. 75-81.

[5] Edwin L. Key, "An analysis of the structure and complexity of nonlinear binary sequence generators," *IEEE Transactions on Information Theory*, Vol. IT-22, No. 6, Nov. 1976, pp. 732-736.

[6] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic application," *IEEE Transactions on Information Theory*, Vol. IT-30, No. 5, Sep. 1984, pp. 776-780.

[7] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Transactions on Computers*, Vol. C-34, Jan. 1985, pp. 81-85.

[8] Willi Meier, and Othmar Staffelbach, "Fast correlation attacks on stream ciphers," *Advances in Cryptology-EUROCRYPT'88*, Lecture Notes in Computer Science, Vol. 330, Springer-Verlag, 1988, pp. 301-314.

[9] Wille Meier, and Othmar Staffelbach, "Nonlinearity criteria for cryptographic functions, " *Advances in Cryptology-EUROCRYPT'89*, Lecture Notes in Computer Science, Vol. 434, Springer-Verlag, 1989, pp. 549-562.

[10] Kencheng Zeng, Chung-Huang Yang, Dah-Yea Wei, and T.T.N. Rao, "Pseudorandom bit generators in stream-cipher cryptography," *Computer*, Vol. 24, Feb. 1991, pp. 8-17.

[11] W. T. Penzhorn, "Correlation attacks on stream ciphers: computing low-weight parity checks based on error-correcting codes," *Fast Software Encryption, FSE'96*, Lecture Notes in Computer Science, Vol. 1039, 1996, pp. 145-158.

[12] W. T. Penzhorn, "Correlation attacks on stream ciphers," *AFRICON, 1996., IEEE AFRICON 4th*, Vol. 2, 1996, pp. 1093-1098.

[13] Solomon W. Golomb, *Shift register sequences*, Holden-Day, 1967.

[14] Douglas R. Stinson, *Cryptography : theory and practice*, CRC Press, 1995.

[15] Bruce Schneier, *Applied cryptography second edition : protocols, algorithms, and source code in C*, John Wiley & Sons, 1996.

[16] Alfred Menezes, Paul van Oorschot, and Scott Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.

[17] Thomas W. Cusick, Cunsheng Ding, and Ari

Renvall, *Stream ciphers and number theory*, Elsevier Science B.V., 1998.