PingPong-128, A New Stream Cipher for Ubiquitous Application

HoonJae Lee, Kevin Chen
Dept. Information Network Eng., Dongseo University, Busan, Korea
ISI, Queensland Univ. of Technology, Brisbane, Australia
E-mail: hjlee@dongseo.ac.kr, chenk@isrc.qut.edu.au

Abstract

The PingPong family of keystream generator is based on the LM-type summation generator. A mutual-clock-control mechanism is added to the LM-type summation generator to provide a security enhancement. PingPong-128, a specific cipher from the PingPong family, is proposed. It takes a 128-bit key and a 128-bit initialisation vector, has 257 bits of internal state, and achieves a security level of 128 bits. In this paper, we present the security analysis of PingPong-128, including the resistance to known attacks against the summation generator and other clock-controlled generators.

1. Introduction

The two basic methods for encrypting text into ciphertext are stream and block ciphers. Stream ciphers (as the name suggests) encrypt text bit-by-bit and are fairly rare with only a few examples in commercial applications such as RC4 [1]. The advantage of a stream cipher is that it is faster and much more efficient than block ciphers. For example, RC4 is close to twice as fast as the nearest block cipher and can be written in 30 lines of code whereas the typical block cipher algorithm takes several hundred lines of code, making them ideal for Internet applications like SSL were speed and efficiency is more valuable [2].

The summation generator [3] was proposed in 1985, and correlation attacks on it were published in [4, 5]. In [6], a fast correlation attack on the summation generator is described. The LM generator [7] was proposed in 2000 as an improvement to the summation generator. The proposed improvement is the addition of an extra memory bit to the combining function. The summation generator succumbs to a range of divide and conquer attacks, from the straightforward divide and conquer attack described in [5, 8], to correlation and fast correlation attack in [4, 5] and [6], respectively.

In this paper, we propose a new generator, PingPong, based on the summation generator, with the addition of a mutual clock control structure. The purpose of the mutual clock control structure is to introduce irregular clocking of the underlying LFSRs thus implicitly increasing the nonlinearity of the output keystream. In other words, The PingPong generator is a clock controlled generator. It is based on the LM generator [7], which is a modification of a summation generator [3] based on two linear feedback shift registers (LFSRs). We demonstrate that the modification defeats the known attacks against the summation generator, and other attacks such as attacks on irregularly clocked keystream generators. An initialisation and rekeying process for the PingPong generator is also defined.

The PingPong generator extends the LM generator through the use of irregularly clocked underlying shift registers.

2. Summation-like Generator

2.1. Description of the Summation Generator

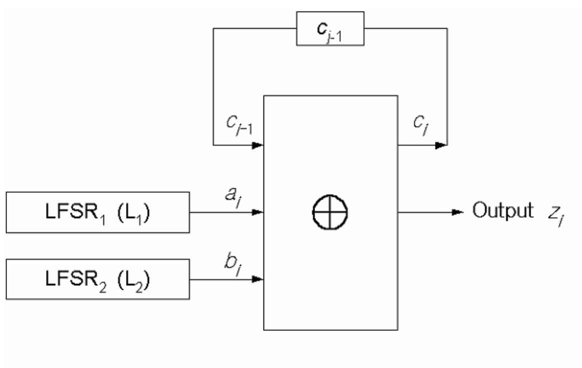

Fig. 1. Summation Generator($r = 2$)

The summation generator uses r regularly clocked binary LFSRs and $Log_2 n$ bits of carry. The LM generator is based on a summation generator with $r = 2$. Denote the two LFSRs L_1 and L_2, respectively, and the carry bit is denoted by c. At time j, denote the output of L_1 as a_j, the output of L_2 as b_j, and the output of f_c as c_j, as shown in fig. 1. The initial state of the carry bit,

c_{-1}, is defined to be 0. At time j, the output of the function f_z is the keystream bit, and is denoted by z_j. The outputs of functions f_c and f_d at time are defined as:

$$c_j = f_c = a_j b_j \oplus (a_j \oplus b_j) c_{j-1} \tag{1}$$

$$z_j = f_z = a_j \oplus b_j \oplus c_{j-1} \tag{2}$$

2.2. Cryptanalysis of the Summation Generator

There are several ways to recover the initial state of the summation generator. A simple approach is the divide and conquer attack [5]. Alternatively, a fast correlation attack [6] could be performed.

2.3. Description of the LM Generator

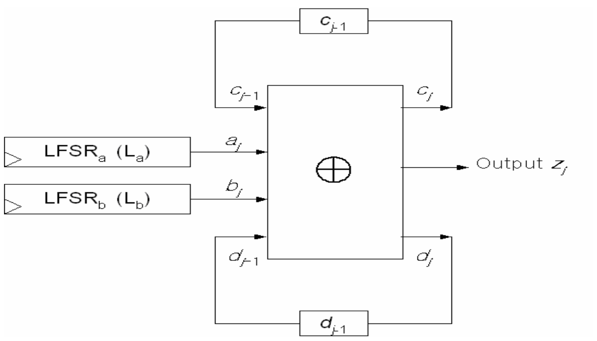

Fig. 2. LM Generator(r = 2)

The LM generator, shown in fig. 2 is very similar to the summation generator, in that L_a, L_b and c are defined in exactly the same way. Another bit of memory, d, is added to the combining function, in an attempt to overcome some of the published attacks on the summation generator. The carry bit, c, is defined by equation 1, identical to the summation generator. The additional memory bit d is calculated by the function f_d and the output function f_z is changed to include d. The value of d_{-1} is defined to be 0.

$$d_j = f_d = b_j \oplus (a_j \oplus b_j) d_{j-1} \tag{3}$$

$$z_j = f_z = a_j \oplus b_j \oplus c_{j-1} \oplus d_{j-1} \tag{4}$$

2.4. Cryptanalysis of the LM Generator

Due to the similarity in construction between the summation generator and the LM generator, similar algorithms can be used to attack both generators. These are outlined below[14].

Divide and Conquer Attack The attack on the summation generator given in [5] can be adapted to recover the initial state of the LM generator. The attacking algorithm is given below[14]:

1. Guess the initial state of L_a and carry bits c_{-1} and d_{-1}
2. Set $j = 0$
3. Calculate b_j, bit j of R_b, using equation 2 and the known keystream bit z_j
4. Calculate c_j using equation 1 and the calculated b_j
5. Calculate d_j using equation 3 and the calculated b_j
6. Increment j, if $j < k$ then goto step 3
7. Initialise the LM generator with the guessed initial state of L_a and the calculated initial states L_b, c_{n-1}, and d_{n-1}
8. Produce a candidate keystream sequence $\{z'_j\}_{j=n}^{k}$ and compare with observed keystream sequence $\{z_j\}_{j=n}^{k}$
9. If $\{z'_j\}_{j=n}^{k}$ and $\{z_j\}_{j=n}^{k}$ are identical, then the correct initial states of L_a and L_b are successfully recovered, else go to step 1

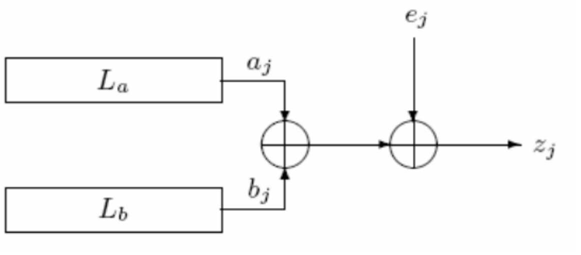

Fig.3. Fast Correlation Attack Model for the LM Generator

This attack requires exhaustive search of m+2 bits, that is, the size of L_a and the carry bit c and the memory bit d, to recover the initial states of both registers, that is, the $m+n$ bits of initial state. Under this attack, the addition of d offers only one extra bit of security over the summation generator.

Fast Correlation Attack The LM generator can be modelled as the modulo-two sum of two LFSRs, plus some binary noise, as shown in fig. 3, where e_j is the noise and z_j is the output keystream bit. For the LM generator, the noise is provided by the modulo-two sum of the carry bit c and the memory bit d. The two memory bits are highly correlated, with $P(c_j = d_j) = 0.75$. Therefore, modelling the LM generator this way, the noise level is 0.25. This is a significant deviation from 0.5, and makes the LM generator vulnerable to a fast correlation attack, similar to the fast correlation attack on the summation generator.

Table 1. Distribution of c_j and d_j in the LM Generator

a_j	b_j	c_{j-1}	d_{j-1}	c_j	d_j
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

The carry bit, *c*, and the memory bit, *d*, are identical with a probability of 0.75, as shown in Table 1. Recall that the output of the LM generator is defined as

$$\begin{aligned} z_j &= a_j \oplus b_j \oplus c_{j-1} \oplus d_{j-1} \\ &= (a_j \oplus b_j) \oplus (c_{j-1} \oplus d_{j-1}) \end{aligned}$$

Since $c_{j-1} \oplus d_{j-1} = 0$ is true with probability 0.75, $z_j = a_j \oplus b_j$ is also true with probability 0.75. This can be exploited in a fast correlation attack to recover the initial states of the LM generator [7].

3. PingPong Generator

3.1. Description of the PingPong Generator

Proposed PingPong family generators are simple, easy to implement in hardware and in software, and high secure. PingPong family in fig.4 is a hybrid generator, combining the LM generator (improved summation generator) with a high secure clock-controlled generator. LFSR A is clock-controlled by function f_b, it has a random integer output. And LFSR B is clock-controlled by function f_a, it also has a random output. Two clock-controlled functions give a multiple clock to the other LFSR. It makes that the output should be more unpredictable. Pingpong Family generator outputs z_j, c_j and c_j from each LFSR outputs a_j and b_j, previous carry c_{j-1} and previous memory d_{j-1} as in fig.4.

$$z_j = f_z = y_j \oplus d_{j-1} \tag{5}$$

$$d_j = f_d = f(a_j, b_j, d_{j-1}) = b_j \oplus (a_j \oplus b_j) d_{j-1} \tag{6}$$

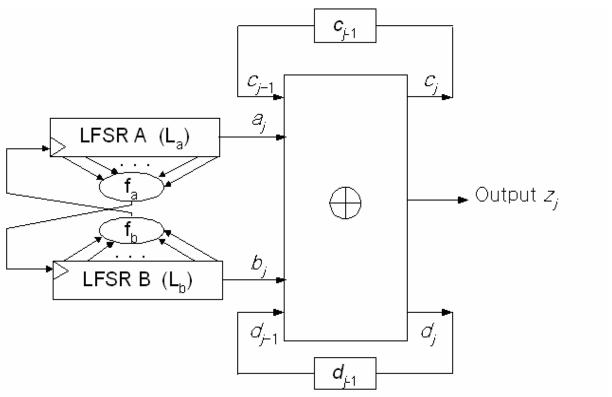

Fig.4. PingPong Family Generator

where (*y*) is the output sequence of summation generator, (*a*) the output sequence of LFSR 1, (*b*) the output sequence of LFSR 2, (*c*) carry sequence, $c_{-1} = 0$ carry initialization value, (*d*) memory sequences, $d_{-1} = 0$ memory initialization value.

3.2. PingPong-128

In this Section, we describe in detail PingPong-128, an instance from the PingPong family of stream ciphers. It has two mutually clocking LFSRs and a single memory bit. The LFSRs are of lengths 127 bits and 129 bits. Together with the memory bit they give PingPong-128 an internal state of 257 bits. PingPong-128 takes a 128-bit key and a 128-bit initialisation vector to fill the internal state.

Keystream Generation The PingPong generator produces the output keystream by combining the LFSR sequences and the memory sequence. PingPong-128 has two mutually clocking LFSRs L_a and L_b, and a single bit of memory c. Two primitive polynomials, $P_a(x)$ and $P_b(x)$ are following:

$$p_a(x) = x^{127} \oplus x^{109} \oplus x^{91} \oplus x^{84} \oplus x^{73} \oplus x^{67} \oplus x^{66} \oplus x^{63} \oplus x^{56} \oplus x^{55} \oplus x^{52} \oplus x^{48} \oplus x^{45} \oplus x^{42} \oplus x^{41} \oplus x^{37} \oplus x^{34} \oplus x^{30} \oplus x^{27} \oplus x^{23} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{16} \oplus x^{13} \oplus x^{12} \oplus x^{7} \oplus x^{6} \oplus x^{2} \oplus x^{1} \oplus 1$$

$$p_b(x) = x^{129} \oplus x^{125} \oplus x^{121} \oplus x^{117} \oplus x^{113} \oplus x^{109} \oplus x^{105} \oplus x^{101} \oplus x^{97} \oplus x^{93} \oplus x^{89} \oplus x^{85} \oplus x^{81} \oplus x^{77} \oplus x^{73} \oplus x^{69} \oplus x^{65} \oplus x^{61} \oplus x^{57} \oplus x^{53} \oplus x^{49} \oplus x^{45} \oplus x^{41} \oplus x^{37} \oplus x^{33} \oplus x^{29} \oplus x^{25} \oplus x^{21} \oplus x^{17} \oplus x^{13} \oplus x^{9} \oplus x^{5} \oplus 1$$

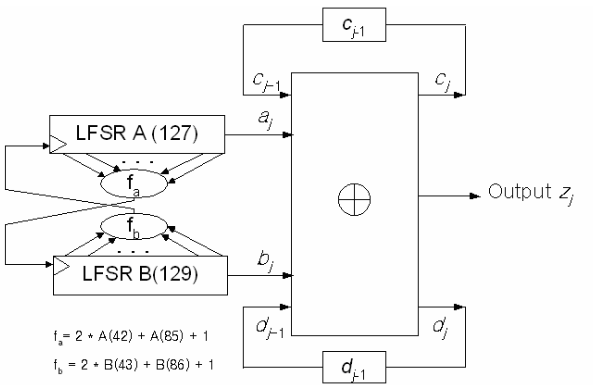

Fig. 5. PingPong-128 Generator

Two clock-control functions, $f_a(L_a)$ and $f_b(L_b)$, and the output keystream bit z and memory bit c at time j are defined to be identical to the summation generator:

$$f_a(L_a) = 2L_{a42}(t) + L_{a85}(t) + 1 \qquad (7)$$

$$f_b(L_b) = 2L_{b43}(t) + L_{b86}(t) + 1 \qquad (8)$$

$$z_j = y_j \oplus d_{j-1} \qquad (9)$$

$$d_j = f(a_j, b_j, d_{j-1}) = b_j \oplus (a_j \oplus b_j) d_{j-1} \qquad (10)$$

Clock Control For PingPong-128, both LFSRs are irregularly clocked, with each register controlling the clocking of the other. Two taps are taken from La to calculate a value in the range 1 . . . 4, and L_b is clocked 1 to 4 times according to this value. Similarly, a value is calculated from two taps taken from L_b to clock L_a. The clock control is calculated by above two functions, f_A and f_B . This clocking scheme can be applied to the PingPong family of keystream generators with n underlying LFSRs, where L_j is used to clock L_{j+1} and L_1 is clocked by L_n.

Key Loading and Rekeying In some communication systems, errors occur which require that the entire message be resent. When a synchronous stream cipher is used, then security requires that a different keystream sequence be used. To achieve this, the rekeying of a stream cipher should include a method for reinitialisation using both the secret key and an additional initialisation vector which is sent in the clear, or otherwise publicly known. We now describe a proposed method for the initial key loading and for the rekeying of PingPong-128. For PingPong-128, both k and iv have a length of 128 bits, and together they fill 257 bits of internal state. The initialisation process can also be used for rekeying. The process to generate the initial state for the keystream generator uses the generator itself twice. The starting state of L_a is obtained simply by XORing the two 128-bit binary strings of the key, k, and iv, that is, $L_a = (k \oplus iv) \bmod 2^{127}$. The starting state of 129 bits for L_b is obtained by considering the 128-bit key, embedded in a 129-bit word and shifted 1 bit to the left, and XORing that with the initialisation vector embedded in a 129-bit word with a leading zero, that is, $L_b = (k{<<}1) \oplus (0|iv)$. Now the cipher is run to produce an output string of length 257 bits. For the second iteration of the cipher, the first 128 bits of this output string are used to form the initial state of La, and the remaining 129 bits are used to form the initial state of L_b. The cipher is run a second time to produce an output string of length 257 bits. The output from this second application is used to form the initial state of the keystream generator when we begin keystream production. As previously, the first 128 bits form the initial state of L_a, and the remaining 129 bits form the initial state of L_b. It is very unlikely that either LFSR will be initialised with the all zero state. By employing the PingPong algorithm itself, we take advantage of both the known security properties of the algorithm and also its fast implementation. Due to the high security of PingPong we conclude that the best attack in the rekeying scenario is exhaustive key search.

Implementation Issue Both LFSRs in PingPong-128 use the Galois implementation rather than the Fibonacci implementation. This is a design decision based on the software performance of the implementation. It is observed that the Galois implementation is much more efficient in software than the Fibonacci, although both implementations are equally efficient in hardware. It is worth noting that these two implementations give different output sequences with the same initial LFSR states, therefore it is essential to specify the style of implementation.

4. Analysis of the PingPong Generator

In this Section, we present the keystream properties of the PingPong generator based on empirical results.We also show the resistance of the PingPong generator to known attacks.

4.1. Keystream Properties

There are three basic requirements for the pseudo-random binary sequences: long period, high linear complexity, and good statistical properties. Long period avoids the keystream to be reused when encrypting long messages. High linear complexity prevents attacks using the Berlekamp-Massey algorithm [12]. Good statistical properties guard against attacks exploiting the biases in the keystream. Experiments have been done on several instances from the PingPong family of keystream generators to observe the keystream properties of PingPong. Each instance of PingPong has a pair of LFSRs of different lengths. For each pair, we used a number of different feedback polynomials and took clocking taps from various stages of the registers. It was observed that the choice of feedback polynomials and clocking tap position did not influence the keystream properties. For each pair of LFSR lengths, 50 random initial states were used to run the experiment. The results of the experiments varied widely, for example, for register lengths 9 and 10, the linear complexity varied between 400 and 822. The lowest resulting linear complexity and shortest period of the experiments are tabulated in Table 2.

Table 2. PingPong Keystream Properties

Register Lengths	Linear Complexity	Period
5, 6	23	25
5, 7	50	50
6, 7	43	51
7, 8	93	101
8, 9	200	200
9, 10	400	401
10, 11	815	815
11, 13	3276	3276
13, 15	13100	13105

From the empirical results, we derived the following equations for calculating the minimum linear complexity and period. Denote the sum of register lengths n, the lower bound of the linear complexity *LC* can be expressed as

$$LC \geq 25\times 2^{\lceil (n-11)/2 \rceil} = 2^{4.6}\times 2^{(n-11)/2}$$

Similarly, the period *P* can be expressed as

$$P \geq 25\times 2^{\lceil (n-11)/2 \rceil} = 2^{4.6}\times 2^{(n-11)/2}$$

For PingPong-128, n = 256, the lower bound of the linear complexity is therefore

$$LC \geq 2^{4.6}\times 2^{\lceil (256-11)/2 \rceil} = 2^{4.6}\times 2^{123} \approx 2^{128}$$

and the Period *P* is

$$P \geq 2^{4.6}\times 2^{\lceil (256-11)/2 \rceil} = 2^{4.6}\times 2^{123} \approx 2^{128}$$

The design strength of PingPong-128 is 2^{128}. It is therefore resistant against attacks based on basic keystream properties such as linear complexity and period.

4.2. Time Memory Tradeoff Attack

The objective of time-memory tradeoff attacks is to recover the internal state at a known time. The attacks are conducted in two stages. During a preprocessing phase, the cryptanalyst constructs a lookup table, mapping possible internal states to prefixes of the corresponding output keystreams. In the real time phase of the attack, the cryptanalyst takes a segment of known keystream and tries to find the corresponding internal state, by searching through the lookup table.

Let *S, M, T, P* and *D* denote the cardinality of the internal state space, the memory(in binary words of size equal to log_2 *S*), the computational time (in table lookups), the pre-computation time (in table lookups), and the amount of data (without re-keying, this is the length of known keystream), respectively. For the time-memory attacks described in [15] $T \bullet M = S$, $P = M$ and $D = T$. For example, a $2^{128} \bullet 2^{128} = 2^{256}$ tradeoff could be used. Therefore PingPong-256 with 256-bit of internal state can only have 128 bits of security.

The more general time-memory-data tradeoff[16] asserts that $T \bullet M2 \bullet D2 = S2$, $P = S/D$, $D2 = T$. This decreases D at the cost of increasing P. For example, one may choose $M = D = S^{1/3}$ and $T = P = S^{2/3}$, but for PingPong-256, with $S = 256$, this gives $M = D = 2^{85.3}$ and $T = P = 2^{170.7}$, clearly better than exhaustive key search.

4.3. Mutual Irregular Clocking of LFSRs

In this section we consider two LFSRs that clock each other in an irregular fashion. Let L_a and L_b be the two LFSRs with primitive polynomials and length len_a and len_b respectively. When clocked autonomously they produce m-sequences with period $2^{len_a}-1$ and $2^{len_b}-1$ for any non-zero initial state. Now consider the cycle structure for the situation where they clock each other using two bits from each register to select from 1, 2, 3 or 4 clock cycles for the other register to obtain the next state. This is the general model for the PingPong structure.

Let L_a be clocked stepa cycles by the bits $L_b[c1]$ and $L_b[c2]$ and similarly L_b is clocked stepb cycles by $L_a[c3]$ and $L_b[c4]$. The clocking positions c_i are fixed by the algorithm specification, and also

$$step_a = 2\times L_b[c1] + L_b[c2] + 1$$
$$step_b = 2\times L_a[c3] + L_a[c4] + 1$$

Clearly $step_a$ and $step_b$ are in the set {1,2,3,4}.
Now define the cumulative clocking values

$$SUM_a[t] = \sum_{i=0}^{t} step_a[i]$$

And similarly

$$SUM_a[t] = \sum_{i=0}^{t} step_a[i]$$

Then the state of the system at time t is given by

$$[L_a[SUM_a[t]], L_b[SUM_b[t]]]$$

Now consider how

$$[L_a[SUM_a[t-1]], L_b[SUM_b[t-1]]]$$

Evolves into

$$[L_a[SUM_a[t]], L_b[SUM_b[t]]]$$

Any state could have up to four precursor states, corresponding to $step_a$ in {1,2,3,4}. Consider the precursor state associated with $setp_a = i$, then we have

$$SUM_a[t-1] + i = SUM_a[t]$$

Clearly there must also be some value for $step_b = j$. Noting that the values for i and j are specified by the bits in the registers at time $t-1$.
Clearly, in order to obtain state

$$[L_a[SUM_a[t]], L_b[SUM_b[t]]]$$

from the previous state with clocking of (i,j), we must have both

$$i = step_a = 2\times L_b[SUM_b[t-1]][c1] + L_b[SUM_b[t-1]][c2] + 1$$

And

$$j = setp_b = 2\times L_a[SUM_a[t-1]][c3] + L_a[SUM_a[t-1]][c4] + 1$$

Where both

$$SUM_a[t-1] + i = SUM_a[t]$$

And

$$SUM_b[t-1] + j = SUM_b[t]$$

Given any state, there are 16 bits (4 bits after each of the 4 clocking taps) that could have influenced the progression to that state. Four checks of the above expressions gives the means to determine how many precursor states exist. Note that there will be states that are unreachable (the have no precursor states), and these are the states that exist as the starts of trails leading to cycles. The next-state diagram is more comparable to that of random functions, rather than random bijections.

Although more precise work needs to be done in the analysis and security comparison of the PingPong style structure, it seems clear that it does not produce the same quality state sequences as an LFSR of the same size.

5. Conclusion

In this paper, we have proposed PingPong, a generator based on the summation generator with a mutual clock control structure. It defeats known attacks against the summation generator and other clock controlled keystream generators.

6. Acknowledgement

This research was supported by University IT Research Center Project, and by the Program for Training of Graduate Students in Regional Innovation.

7. References

[1] A.J. Menezes, P.C. Oorschot and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[2] Wedbush Morgan Securities - Industrial Report, "Access Management/Internet Security Industry," on http://www.vikasqupta.com, Feb. 28, 2002.

[3] R. A. Rueppel, "Correlation Immunity and the Summation Generator," Advances in Cryptology, Proceedings of CRYPTO'85, pp. 260-272, 1985.

[4] W. Meier and O. Staffelbach, "Correlation Properties of Combiners with Memory in Stream Ciphers," Advances in Cryptology, Proceedings of EUROCRYPT90, pp. 204-213, 1991.

[5] E. Dawson, "Cryptanalysis of Summation Generator," Advances in Cryptology - AUSCRYPT'92, Lecture Notes in Computer Science, Springer-Verlag, pp. 209-215, 1993.

[6] J. Golic, and M. Salmasizadeh and E. Dawson, Fast Correlation Attacks on the Summation Generator," Journal of Cryptology, Vol. 13, No. 2, pp.245-262, 2000.

[7] Hoonjae Lee, Sangjae Moon, "On An Improved Summation Generator with 2-Bit Memory," Signal Processing, 80(1), pp. 211217, Jan. 2000.

[8] T. Siegenthaler, "Design of Combiners to Prevent Divide and Conquer Attacks,"Advances in Cryptology, Proceedings of CRYPTO'85, pp. 273-279, 1985.

[9] R. A. Rueppel, Analysis and Design of Stream Ciphers, Springer-Verlag, 1986.

[10] W. Meier and O. Staffelbach, ”Correlation Properties of Combiners with Memory in Stream Ciphers,” Journal of Cryptology, Vol. 5, pp. 67-86, 1992.

[11] A. Clark, E. Dawson, J. Fuller, J. Golic, Hoon-Jae Lee, W. Millan, Sang-Jae Moon, L. Simpson, ”The LILI-II Keystream Generator,” LNCS 2384, pp.25-39, Jul. 2002 (ACISP’2002).

[12] J. L. Massey, ”Shift-Register Synthesis and BCH Decoding,” IEEE Trans. on Infor. Theo., Vol. IT-15, No. 1, pp. 122-127, Jan. 1969.

[13] R. A. Rueppel and O. J. Stafflebach, ”Products of Linear Recurring Sequences with Maximum Complexity,” IEEE Trans. on Infor. Theo., Vol. IT-33, No. 1, pp. 124-131, Jan. 1987.

[14] Kevin Chen, Ed.Dawson, etc. ”Security Analysis of the LM Generator,” Report, Aug. 2004.

[15] S. Babbage, ”Improved exhaustive search attacks on stream ciphers,” European Convention on Security and Detection, Vol. 408, pp. 161-166, May 1995.

[16] A. Biryikov and A. Shamir, ”Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers”, Advances in Cryptology, Proceedings of ASIACRYPT00, LNCS 1976, pp.1-13, 2000.