Improved Cascaded Stream Ciphers Using Feedback

Lu Xiao[1], Stafford Tavares[1], Amr Youssef[2], and Guang Gong[3]
[1]*Department of Electrical and Computer Engineering, Queen's University, {xiaolu, tavares}@ee.queensu.ca*
[2]*Concordia Institute for Information Systems Engineering, Concordia University, youssef@ciise.concordia.ca*
[3] *Department of Electrical and Computer Engineering, University of Waterloo, ggong@calliope.uwaterloo.ca*

Abstract

Currently, there are no stream ciphers which have been adopted as international standards. Most stream ciphers in practice were designed based on linear feedback shift registers. However, most of these ciphers have been attacked. This paper examines the security of a family of stream ciphers called Cascade Stream Ciphers (CSCs) and proposes two modified versions of this type of cipher. A cascade stream cipher consists of a number of small RC4 cells that are cascaded to generate an output sequence (the keystream). Each RC4 cell is composed of a Substitution-box (or S-box) and two pointers addressed to the S-box. Both the pointers and the S-box contents are updated each clock cycle. Each cell operates internally as a small RC4 cipher. The output of the current cell is used as a pointer of the next cell in the cipher. The CSCs possess several good properties that are required for stream ciphers, such as simplicity, scalability, and efficiency in hardware. It can be used as a stream cipher as well as a pseudorandom number generator.

In a previously proposed CSC, the output of the last cell was directly used as the keystream. Due to this structure, a Backward State Transition (BST) attack, proposed in this paper, is able to break it. The attack is simulated using a combinatorial model and verified by experiments. The attack runs much faster than an exhaustive key search and has a workload linear in the number of cells.

To prevent the BST attack, two modifications of the CSC structure are proposed. In the modified CSC, outputs from two cells are combined to mask the output of the cipher and the output of the last cell is used in a feedback loop to drive one pointer of the first cell. The improved structure is immune to the BST attack and has a cycle distribution closer to that of a random permutation. In particular, the average of cycle lengths is increased and there are fewer of them. The experiments show that the use of larger S-boxes (e.g., 4-bit) in a shorter cascade can improve the throughput of the cipher.

Keywords*: stream cipher, cryptanalysis, symmetric cipher, S-box, GST cipher, RC4*

1. INTRODUCTION

A stream cipher [1,2,3] usually produces ciphertext by mixing the plaintext with the keystream (i.e., a random-looking sequence), typically bit by bit. A classic method for keystream generation is to combine outputs from Linear Feedback Shift Registers (LFSRs). If the correlation between keystrean and internal LFSR information is recognizable, a correlation attack [4,5] can be used to recover part of the LFSRs initial contents. Moreover, an LFSR-based stream cipher could also be vulnerable to an algebraic attack [6].

By avoiding the LFSR structure, Substitution-box (S-box) based ciphers, such as RC4[7], show high resistance to these two attacks. As a widely used stream cipher, RC4 has been studied in [8,9,10,11,12,13] for security evaluation based on different attacks. Recently, a Cascaded Stream Cipher (CSC)[14] structure has been proposed using RC4 structure in each cascaded cell. GST[15] is a CSC cipher with a 2-bit S-box in each cell. Since the cell is very small, GST is simple, scalable, and efficient for hardware realization.

Figure 1 shows a GST cipher with N cascaded cells, denoted as GST-N. In the k-th cell ($1 \le k \le$ N), an S-box S_k has two pointers, denoted as i_k and j_k. Pointers i_k and j_k store possible indices to the mapping table of S-box S_k. The S-box in each cell performs a bijective mapping from a 2-bit index to a 2-bit output. By swapping entries in the mapping table, the S-box can assume any of 24 permutations of {00, 01, 10, 11}. The values of i_k and j_k are updated while determining the output of the k-th cell. The S-boxes are cascaded with the output of S_k connected to the left pointer i_{k+1} of S-box S_{k+1}.

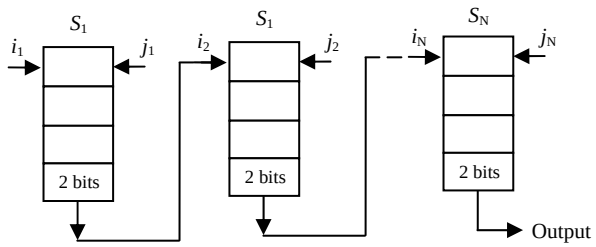

Figure 1. Structure of the GST-N cipher [15]

During keystream generation, the k-th cell performs the following operations:

if k=1 then
$i_1 = i_1 + 1 \bmod 2^2$
else
begin
1. $j_k = j_k + S_k[i_k] \bmod 2^2$
2. Swap $S_k[i_k]$ and $S_k[j_k]$
3. $q = S_k[i_k] + S_k[j_k] \bmod 2^2$
4. if $k < N$, $i_{k+1} = S_k[q]$
else output $S_k[q]$ as the next 2-bit word in the keystream
end

Before keystream generation, i_1 and all j_k pointers ($1 \le k \le N$) are initialized to zero. A key schedule is used to initialize the S-boxes based on a user key. Thus, the keystream sequence can be restored with knowledge of either the user key or S-box contents in all cells.

The cryptographic properties of GST ciphers have been examined in [14,15]. According to cycle lengths, key spacing, and other statistical features, GST is advantageous for use in many applications such as pseudorandom number generators, especially in hardware. A known plaintext attack, called the Reverse Cascade Attack (RCA), was introduced in [15,17], which requires a workload higher than an exhaustive key search.

This paper introduces a new plaintext attack on the GST, which is much faster than an exhaustive key search. The length of the required keystream sequence is linear in the number of cells, N. The effectiveness of this attack on CSCs with 3- and 4-bit S-boxes is also examined. Accordingly, two modifications in the cascaded structure are suggested for security enhancement. Some statistical properties of CSCs are further explored.

2. BST ATTACK

A Backward State Transition (BST) attack is a known plaintext attack to determine the initial states of all GST cells. It exploits the fact that, given a keystream sequence of a CSC such as GST-N, it is possible to remove some candidates of the initial state in the last cell. In the case of GST-N, if the state of the last cell is specified, the left pointer i_N can be determined with knowledge of the keystream. Since the output of the preceding cell is fed into the last cell as i_N, we can launch a similar attack on GST-(N-1).

2.1 Initial State Estimation

The attack begins with a statistical analysis of the initial state of an independent 2-bit cell used in a CSC. We denote the S-box as S and the two pointers as i and j. Suppose the sequence of i is a random input, and the initial state is defined as the state of S and j right before the first output is generated. Since there are 24 permutations for a 2-bit S-box and 4 values for j, we have 96 candidates for an initial state.

Table 1. Initial states associated with (1, 2)

S-box	j=0	j=1	j=2	j=3	S-box	j=0	j=1	j=2	j=3
{0,1,2,3}	0	0	0	1	{2,0,1,3}	1	1	1	0
{0,1,3,2}	0	0	1	1	{2,0,3,1}	1	0	1	1
{0,2,1,3}	1	1	0	0	{2,1,0,3}	0	1	0	0
{0,2,3,1}	0	1	0	0	{2,1,3,0}	0	1	0	0
{0,3,1,2}	1	0	1	1	{2,3,0,1}	0	1	0	1
{0,3,2,1}	0	1	0	0	{2,3,1,0}	1	1	1	0
{1,0,2,3}	1	0	0	0	{3,0,1,2}	1	1	0	1
{1,0,3,2}	1	0	1	0	{3,0,2,1}	1	0	1	1
{1,2,0,3}	0	0	0	1	{3,1,0,2}	0	0	1	1
{1,2,3,0}	0	0	0	0	{3,1,2,0}	0	0	0	1
{1,3,0,2}	0	1	1	0	{3,2,0,1}	1	0	0	1
{1,3,2,0}	1	0	0	1	{3,2,1,0}	0	1	0	1

Suppose two sequential output words of this cell are known as (a_1, a_2), Table 1 shows the possible initial states where a_1=1 and a_2=2. A possible initial state to generate the first output a_1 is marked as "1"; an impossible state is marked as "0". As we can see from the table, only 44 out of 96 states are possible for outputs (1, 2). A similar table can be constructed for any other two outputs. The expected number of initial states for any (a_1, a_2) is 43.5.

2.2 Backward State Transition

The output sequence of the S-box, (a_1 ,…, a_l), can be divided into many two-word pairs in the form of (a_λ , $a_{\lambda+1}$) where $1 \le \lambda \le l$-1. We can get the possible initial states associated with (a_{l-1}, a_l) as illustrated in Table 1. It would be desirable to further reduce the initial states to generate a_{l-2}, based on initial states determined by (a_{l-1}, a_l). The following theorem helps us to achieve this.

Theorem 1: Let a 2-bit RC4 cell perform the following operations

1. $j = j + S[i] \bmod 2^2$
2. Swap $S[i]$ and $S[j]$
3. $q = S[i] + S[j] \bmod 2^2$
4. Output = $S[q]$.

If the output, j, and $S[.]$ at the end of above operations are known, then i, j, and $S[.]$ at the beginning of these operations can be uniquely determined.

Proof: Since $S[.]$ and j are updated in Steps 1 and 2, denote the updated S [.] and j as S_z [.] and j_z. The above operations can be rewritten as

1. $j_z = j + S[i] \bmod 2^2$
2. Swap $S[i]$ and $S[j_z]$

3. $q = S_z[i] + S_z[j_z] \bmod 2^2$

4. Output = $S_z[q]$.

From step 4, we get $q = S_z^{-1}[\text{Output}]$

$\Rightarrow$ $S_z[i] + S_z[j_z] \bmod 2^2 = q$

$\Rightarrow$ $S_z[i] = q - S_z[j_z] \bmod 2^2$

$\Rightarrow$ $i = S_z^{-1}[q - S_z[j_z] \bmod 2^2]$.

From step 2, $S[.]$ can be obtained by swapping $S_z[i]$ and $S_z[j_z]$.

From step 1, $j = j_z - S[i] \bmod 2^2$.

According to the theorem, if we know a unique state of the S-box right before generating a_{l-1}, we can get the corresponding unique state right before generating a_{l-2}. The procedure from initial states determined by (a_{l-1}, a_l) to the corresponding states before generating a_{l-2} is called Backward State Transition (BST) in this paper. After backward state transition, we get the possible initial states associated with three outputs (a_{l-2}, a_{l-1}, a_l). This procedure can be repeated to any (a_{λ}, …, a_l), recursively.

For example, given a sequence of (a_1 ,…, a_7) as (2, 3, 0, 1, 3, 1, 2), the attack begins with the last two outputs. From the last pair (1, 2), we get 44 initial states as shown in Table 1. These are the initial states associated with output a_6 (i.e., "1"). Since a_5 is "3", according to Theorem 1, we can calculate S-box content and j value right before generating a_5 from each possible state in Table 1. Thus, the backward state transition is performed as shown in Table 2.

Table 2. Initial states associated with (3, 1, 2)

S-box	j=0	j=1	j=2	j=3	S-box	j=0	j=1	j=2	j=3
{0,1,2,3}	0	0	0	1	{2,0,1,3}	0	0	0	1
{0,1,3,2}	0	0	0	1	{2,0,3,1}	0	0	1	1
{0,2,1,3}	0	1	0	0	{2,1,0,3}	0	0	0	0
{0,2,3,1}	0	0	1	0	{2,1,3,0}	1	1	0	0
{0,3,1,2}	0	0	1	0	{2,3,0,1}	0	1	0	0
{0,3,2,1}	0	0	0	1	{2,3,1,0}	0	1	1	0
{1,0,2,3}	0	1	1	0	{3,0,1,2}	0	1	0	0
{1,0,3,2}	0	1	0	1	{3,0,2,1}	1	0	0	0
{1,2,0,3}	1	1	1	0	{3,1,0,2}	0	1	1	0
{1,2,3,0}	1	0	0	0	{3,1,2,0}	1	0	0	1
{1,3,0,2}	1	1	0	0	{3,2,0,1}	0	0	1	0
{1,3,2,0}	0	0	0	0	{3,2,1,0}	1	0	1	1

In Table 2, there are only 34 initial states for (3, 1, 2) derived from the 44 states for (1, 2). This is because the backward state transition is an injective mapping. It is possible for two initial states of (1, 2) to be mapped to the same initial state of (3, 1, 2).

Similarly, the backward state transition can be performed from (3, 1, 2) to (1, 3, 1, 2), and so on. Table 3 shows the number of initial states associated with backwardly growing sequences. Given more outputs, we can keep performing backward state transition until the number of possible states converges to 1. From then on, during each backward state transition from one state to its preceding state, value i can also be determined uniquely according to Theorem 1.

Table 3. Initial States Reduction

Output sequence	# of initial states
(1,2)	44
(3, 1, 2)	34
(1, 3, 1, 2)	30
(0, 1, 3, 1, 2)	26
(3, 0, 1, 3, 1, 2)	21
(2, 3, 0, 1, 3, 1, 2)	17

Denote the number of outputs required to determine a unique initial state as β. From 200 experiments, we get an average for β as 141.9. The value of β varied from 31 to 476 with a standard deviation of 74.8.

2.3 The Balls-and-Bins Model

The backward state transition can be modeled by a balls-and-bins problem [16]. Assuming that the backward transition from one state to its predecessor is random, it is equivalent to throwing m balls into n bins where n is the number of all possible states and m is the number of possible initial states at the current moment. Thus, the number of states after backward state transition can be regarded as the number of nonempty bins. Since the number of empty bins is expected to be

$$\text{E[\# empty bins]} = n\,(1 - 1/n)^m,$$

the number of nonempty bins is expected to be

$$\text{E[\# nonempty bins]} = n - n\,(1 - 1/n)^m.$$

In our attack, each backward transition is an injective mapping from m states to $\lceil n - n\,(1 - 1/n)^m \rceil$ previous states. However, when m is small enough, it is possible that $m = \lceil n - n\,(1 - 1/n)^m \rceil$. In this case, we need more than one state transition to make the number of predecessor smaller. This is equivalent to the event of at least 2 balls thrown into the same bin. Such an event has a probability of

$$P = 1 - \frac{n!}{(n-m)!\,n^m}.$$

After $1/P$ steps of backward state transitions, the number of states is expected to decrease. For simplification of analysis, we assume that only one state is deleted after $1/P$ steps. Such an assumption is acceptable because the event of the number of states decreasing by 3 or more after $1/P$ steps happens with a fairly small probability.

In a BST attack on GST, n = 96 and m = 44 (i.e., E[m] = 43.5). The steps can be simulated as Table 4 shows. In Step 13, $m = \lceil n - n\,(1 - 1/n)^m \rceil$. After Step 13, $\lceil 1/P \rceil$ is used to calculate step intervals to make states decrease. Theoretically, it is expected to require 200 sequential outputs to uniquely determine an initial state, thus E[β] $\approx$

200. By summing up the second column of Table 4, we get the workload w as 954 backward transition steps. This is equivalent to running a 2-bit cell for 954 times.

Table 4. BST Attack Simulation

Steps	# states m	# previous states $\lceil n\text{-}n(1\text{-}1/n)^m \rceil$
1	44	36
2	36	31
3	31	27
…	…	…
12	15	14
13	***14***	***14***
14	14	13
16	13	12
…	…	…
104	3	2
200	2	1

2.4 Application to the GST Ciphers

If a short segment, say L words, of plaintext and ciphertext is obtained, we can get the corresponding part of the GST-N keystream by XORing plaintext and ciphertext. Thus, L sequential outputs of the N-th cell are available.

Starting from the last output and working backward, we apply BST until one initial state is uniquely determined. From this state, all i_N can be calculated backward. Because i_N is the output of (N-1)-th cell, the next task is to perform a similar BST attack to GST-(N-1) with a sequence of L-E[β] outputs.

This procedure is repeated until the state of the first cell is determined. Then, we know all information to restore the whole keystream. The expected number of required keystream words is: E[L] = E[β]×N. The workload to break GST-k ($1 \le k \le N$) is composed of both BST workload and computation to determine E[β]×(k-1) values of i_k. Thus, the workload to break the whole cipher is expected to be:

$$W = \frac{1}{N}\sum_{k=1}^{N}(w + E[\beta]\times(k-1)) = w + \frac{N-1}{2}E[\beta]\cdot$$

As deduced from the balls-and-bins model, w = 954 and E[β] = 200. The unit of W is the number of operations to generate one GST-N keystream output. Note that W is linear in N while the workload of an exhaustive key search is exponential in N. In practice, it costs less than 1 second to break GST-32 on a Pentium IV computer, which is much faster than RCA [15,17] and requires only a keystream sequence of approximately 1600 bytes (6400 2-bit words).

2.5 Extension to other CSCs

The BST attack also works for other CSCs, denoted by CSCM-N (Cascaded Stream Cipher with M-bit S-boxes and N cells). Table 5 shows E[β] and w calculated for 3-bit and 4-bit RC4 cells (denoted by RC4/3 and RC4/4, respectively). As we can see, E[β] is comparable to the number of S-box permutations. In theory, a BST attack on CSCs with M-bit S-boxes requires a workload linear in N and exponential in M.

Table 5. Complexity Estimation

Cipher	E[β]	w	# of permutations
CSC3-N	8.10E04	8.93E05	4.03E04
CSC4-N	4.18E13	3.07E14	2.09E13

Note that when S-boxes become larger, the balls-and-bins model is not accurate because the backward transition of a state is not random enough. This is caused by the fact that not all states are accessible as previous states by swapping $S[i]$ and $S[j]$ of current S-box permutation. For example, 7 of 24 permutations are accessible with 2-bit S-boxes, while only 29 of 40,320 permutations are accessible with 3-bit S-boxes. On the other hand, every state is accessible from a long run of BSTs. As a result, the balls-and-bins model just presents a lower bound of the workload when the S-box size is larger than 2 bits. In a real BST attack on CSC3-N, β is approximately 6.4×10^5.

3. Enhanced CSC Structures

Since a BST attack is quite efficient on the CSC, it is necessary to make some changes to frustrate it. One simple way is to make the output of the last cell invisible from the keystream. In addition, a feedback method is suggested to enhance other security properties such as cycle lengths.

3.1 Feedforward

Since a BST attack is quite efficient on the CSC, it is necessary to make some changes to frustrate it. One simple way is to make the output of the last cell invisible from the keystream. In addition, a feedback method is suggested to enhance other security properties such as cycle lengths.

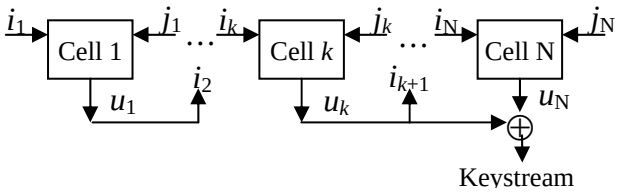

Figure 2. Feedforward

With the output masking as shown in Figure 2, even if

one initial state of Cell N is uniquely identified, i_N is still a function of u_N. Because u_N is not accessible from the keystream, an attacker has to guess a sequence of $E[\beta]\times(N-1)$ words to deduce a unique state for all cells, which is much harder than an exhaustive key search (i.e., $4^{E[\beta]\times(N-1)} > (4!)^N$).

Feedforward is advantageous because such a modification does not affect the internal mechanism of the cipher. As a result, many cryptographic tests discussed in [14,15] are still valid, e.g., the distribution of cycle length and key spacing. By adding (N-k) registers to delay i_{k+1}, the cipher can be easily pipelined in hardware. Although the modification inevitably influences some statistical features such as maximum deviation of keystream, no evident degradation was observed when we did the statistical tests discussed in [17].

3.2 Feedback

In the original design of GST, i_1 is defined to increment each time: $i_1 = i_1 + 1 \bmod 2^2$. In Figure 3, a feedback method is suggested where $i_1 = u_N + 1 \bmod 2^2$. The adder is inserted because a direct feedback without increment may cause extremely short cycles. We conjecture that all cell outputs and pointers in a feedback mode shown in Figure 3 have a same cycle length. Although a complete proof is still open, this conjecture was supported by our experimental observations.

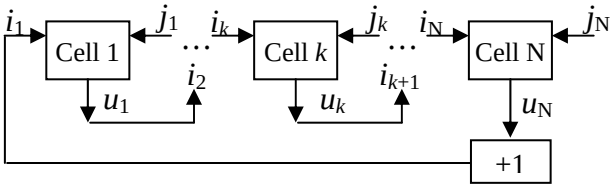

Figure 3. Feedback with Increment

When a CSC adopts the above feedback method, the longest cycle occupies a larger percentage of states, which is desirable because more user keys are then mapped to the different starting points in the longest cycle. Table 6 shows the distribution of cycles in the original CSCs and the modified CSCs as shown in Figure 3. In the case of feedback CSC3-2, the seven longest cycles occupy 99.770% of states and 99.772% of keys. It should be noted, however, that the feedback method is not suitable for a pipelined hardware implementation due to data dependency between i_1 and u_N .

Table 6. Distribution of Cycles

Cipher	# cycles	# states in longest cycle	% states in longest cycle
GST-2	10	14,596	39.6
GST-3	58	788,184	22.3
GST-4	292	29,950,992	8.8
GST-5	1,512	1,430,699,920	4.4
Feedback GST-2	4	33,521	90.9
Feedback GST-3	8	3,482,541	98.4
Feedback GST-4	12	237,802,414	70.0
Feedback GST-5	19	16,011,229,440	49.1
CSC3-1 (RC4/3)	38	955,496	37.0
Feedback CSC3-1	8	1,773,404	68.7
Feedback CSC3-2	>13	571,553,344,008	68.7

Feedback and feedforward can be applied together in order to achieve higher randomness and security as suggested in Figure 4. For example, a feedback CSC4-3 with feedforward output (k = 1) and a feedback CSC3-9 both pass the 16 NIST randomness tests [18]. From the trend illustrated by Table 6, it is reasonable to suggest that CSC4-3 can be used as a simple but very good pseudorandom number generator or keystream generator.

4. CONCLUSION

This paper has presented a BST attack to break the original version of the CSC. The attack workload was evaluated by the balls-and-bins model and verified by our experiments. The attack is efficient since the workload increases linearly with the number of cells in the cipher. A BST attack can be easily frustrated by combining outputs. As a simple solution, a feedforward method was suggested without any change to the internal mechanism of the CSC. To get a better cycle distribution, a feedback method was described.

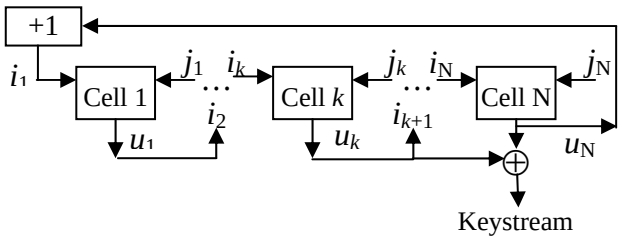

Figure 4. Suggested Enhancement

References

[1] A. Menezes, P. van Oorschot, and S. Vanstone, The Handbook of Applied Cryptography, CRC Press, 1996.

[2] A.Biryukov, Block Ciphers and Stream Ciphers: The State of the Art, LNCS, to appear in Proc. of the COSIC Summer course 2003, Springer-Verlag.

[3] R.A. Rueppel, Good stream ciphers are hard to design, Proc. of International Carnahan Conference on Security Technology , pp. 163-174, 1989.

[4] W. Meier and O. Staffelbach, Fast Correlation Attacks on Certain Stream Ciphers, J. of Cryptology, Vol. 1, pp159-167, 1989.

[5] M. Zhang, Maximum Correlation Analysis of Non-linear Combining Functions in Stream Cipher, J. of Cryptology, Vol.13, pp. 301-313, 2000.

[6] N. Courtois and W. Meier, Algebraic Attacks on Stream Ciphers with Linear Feedback, Eurocrypt'03, LNCS 2656, pp. 345-359, 2003.

[7] R. Rivest, The RC4 Encryption Algorithm, RSA Data Security, Inc., March 1992.

[8] S. R. Fluhrer and D. A. McGrew, Statistical Analysis of the Alleged RC4 Stream Cipher, FSE'00, LNCS 1978, pp. 19-30, Springer-Verlag, 2000.

[9] J. D. Golic, Linear Statistical Weakness of Alleged RC4 Keystream Generator, Eurocrypt'97, LNCS 1233, pp. 226-238, Springer-Verlag, 1997.

[10] L. R. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege, Analysis Methods for (Alleged) RC4, ASIACRYPT'98, LNCS 1514, pp. 327-341, Springer-Verlag, 1998.

[11] I. Mantin and A. Shamir, Practical Attack on Broadcast RC4, FSE'01,LNCS 2355, pp. 152-164, Springer-Verlag, 2001.

[12] S. Mister and S. E. Tavares, Cryptanalysis of RC4-like Ciphers, SAC'98, LNCS 1556, pp. 131-143, Springer-Verlag, 1999.

[13] P. Souradyuti and B. Preneel, Non-fortitous RC4 Key Stream Generator, INDOCRYPT'03, LNCS 2904, pp. 318-325, Springer-Verlag, 2003.

[14] L. Gan, S. Simmons and S. Tavares, A New Family of Stream Ciphers Based on Cascaded Small S-Boxes, IEEE Can. Conf. on Elect. and Computer Eng., CCECE 2001, Toronto, Ontario, May, 2001.

[15] I. Lee, S. Simmons and S. Tavares. Cryptanalysis of the GST Stream Cipher, IEEE Can. Conf. on Elect. and Computer Eng., CCECE 2003, Montreal, Canada, May 4-7, 2003.

[16] J. Kinney, Probability: An Introduction with Statistical Applications, John Wiley & Sons, 1997.

[17] I. Lee, Security Properties of the Cascade Stream Cipher. M.Sc. Thesis, Queen's University, Kingston, Ontario, 2003.

[18] A. Rukhin et al., A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22, 2001. Available at csrc.nist.gov/ publications/nistpubs.