ISSN: 0067-2904

Robust and Efficient Dynamic Stream Cipher Cryptosystem

Abdullah Ayad Ghazi*[1], Faez Hassan Ali[2]
[1]Department of Mathematics, College of Science, University of Baghdad, Baghdad, Iraq
[2]Department of Mathematics, College of Science, Al-Mustansiryah University, Baghdad, Iraq

Abstract

In this paper a new technique based on dynamic stream cipher algorithm is introduced. The mathematical model of dynamic stream cipher algorithm is based on the idea of changing the structure of the combined Linear Feedback Shift Registers (LFSR's) with each change in basic and message keys to get more complicated encryption algorithm, and this is done by use a bank of LFSR's stored in protected file and we select a collection of LFSR's randomly that are used in algorithm to generate the encryption (decryption) key.

We implement Basic Efficient Criteria on the suggested Key Generator (KG) to test the output key results. The results of applying BEC prove the robustness and efficiency of the proposed stream cipher cryptosystem.

Keywords: Stream cipher, Dynamic stream cipher, Linear Feedback Shift Registers.

نظام تشفير انسيابي ديناميكي رصين وكفوء

عبدالله اياد غازي*[١]، فائز حسن علي[٢]
[١]قسم الرياضيات، كليه العلوم، جامعة بغداد، بغداد، العراق
[٢]قسم الرياضيات، كليه العلوم، الجامعة المستنصرية، بغداد، العراق

خلاصة

في هذا البحث سيتم تقديم تقنية جديدة تعتمد على خوارزمية تشفير انسيابي ديناميكية. النموذج الرياضي لخوارزمية التشفير الانسيابي الديناميكية تعتمد على فكرة تبديل تشكيلة المسجلات الزاحفة ذات التغذية الراجعة المشتركة بالمنظومة مع اي تبديل للمفتاح الاساسي ومفتاح الرسالة للحصول على خوارزمية اكثر تعقيد، وهذا يتم من خلال استخدام بنك يحوي مجموعة مسجلات زاحفة والتي يتم اختيار بعضها عشوائيا لكي تستخدم في الخوارزمية لتوليد مفتاح التشفير (الحل). في هذا البحث قمنا بتنفيذ مقاييس الكفاءة الاساسية على مولد المفاتيح المقترح لاختبار نتائج المفتاح المخرج. ان نتائج تطبيق تلك المقاييس اثبتت رصانة وكفاءة نظام التشفير الانسيابي المقترح.

1. Introduction

A stream cipher is a symmetric cipher which operates with a time-varying transformation on individual plaintext digits. The most important stream cipher is the Vernam cipher, and it is also called one-time pad that leads to good secrecy (the ciphertext gives no information about the original plaintext). In general, the stream ciphers have some advantages that make them suitable for many applications. As usual, they are faster and have a lower hardware complexity than the block ciphers. They are also very suitable when the memory is limited, since the digits (or bits) are individually encrypted (decrypted). Moreover, synchronous stream ciphers are not affected by error-propagation. In the stream cipher, the running output key, also called the key stream, is the output sequence which is combined, digit by digit, to the sequence obtained from plaintext to obtain the ciphertext sequence.

*Email: faezhassan@uomustansiriyah.edu.iq

The output running key is generated by a finite state automaton called the key generator (KG) or the key stream generator [1].

Stream ciphers have a history that has been improving throughout the upcoming years. The stream ciphers were invented in 1917 by Vernam G., even though they were not called stream ciphers back in the day [2]. In 1967, Golomb S., an American mathematician and professor had written a book that is an excellent introduction to the theory of both linear and nonlinear shift registers [3]. Rivest R., from RSA Data Security, Inc., in 1987, made a design of a byte oriented stream cipher called RC4 [4]. This kind of cipher found its application in many Internet and security protocols. In 1999 the exact design of both A5/1 and A5/2 was discovered by Briceno [5]. The security of GSM conversation is based on usage of the A5 family of stream ciphers. Many hundred million customers just in Europe are protected from over-the-air piracy by the stronger version of the stream cipher, the A5/1 stream cipher while the other customers on other markets use A5/2 which is weaker version. In 2002, Scream was developed by Coppersmith, Halevi, and Jutla who are the IBM researchers [6]. This cryptosystem is a purely software-oriented stream cipher. The design of Scream is based on the ideas behind the SEAL stream cipher, but considered to be more secure. The "toy cipher" which denoted Scream0 uses the AES whereas the scream cryptosystem cipher uses secret S-boxes, generated by the output key. Hell, Johansson T., and Meier W., and was especially designed for being very small and fast in hardware implementation. This cryptosystem uses the key of length 80 bits and the IV is 64 bits, while its internal state is of size 160 bits. Grain uses a LFSR and a nonlinear FSR (NLFSR), and the idea to use NLFSR is quite new in modern cryptography [7]. At the conference FSE 2004, a new stream cipher cryptosystem called VMPC [8] was proposed by the researcher Zoltak B., which considered to be a modification of the RC4 stream cipher. Dragon is a stream cipher cryptosystem [9] submitted to the eSTREAM project, designed by a group of researchers, Ed Dawson et al. It is a word oriented stream cipher that acts on key sizes of 128 and 256 bits. The original idea of the design is to use a two parts; NLFSR and a linear part.

In this paper we will discuss a new techniques based on dynamic stream cipher algorithm. This algorithm depends on the basic key (BK) and the message key (MK), when both keys change the structure of the LFSR would change too. The generated key gets tested by Basic Efficiency Critter (BEC) and the results of BEC are not fixed, they are changed when the structure of the proposed generator is changed. If the generated key passes all the (BEC) for many times then the generator is secure and ready to be used in encryption and decryption process. In section 2, the stream cipher based on LFSR's will be discussed. In section 3, the Basic Efficiency Criteria for stream ciphers will be described, in section 4, the new design of Dynamic stream ciphers will be introduced. Lastly in section 5, the results of the implementation of BEC on RDSCC will be shown.

2. Stream Cipher Based on Linear Feedback Shift Register

A feedback shift register (FSR) is made up of two main parts: a shift register and its feedback function. The SR is a sequence of bits, (the length of a SR is figured in bits). Each time only one bit is needed, all the bits in the SR are shifted 1 bit to the right.

Cryptographers have liked stream ciphers made up of SR: Since they are easily implemented in digital hardware. Selmer E., the Norwegian governments' chief cryptographer, worked out the theory of SR sequences in 1965 [10]. Golomb S., an NSA mathematician, wrote a book with Selmers results and some of his own [3, 10]. The simplest kind of FSR is a **Linear Feedback Shift Register** (LFSR), as described in Figure-1. The feedback function (polynomial) is simply the XOR of certain stages in the register. Because of the simple feedback sequence, a large background of mathematical theory can be applied to analyzing LFSRs.

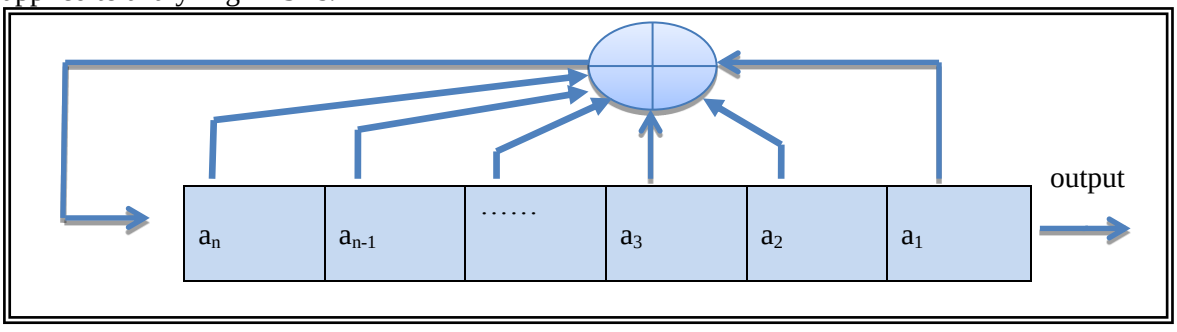

Figure 1- Linear Feedback Shift Register (LFSR).

3. Basic Efficiency Criteria (BEC) for Stream Cipher

The basic efficiency criteria are discussed in the following subsections.

3.1 Randomness [11, 3]

A random bit generator is a device or an algorithm which its outputs sequence (of bits) of statistically independent and unbiased binary digits. A Pseudo Random Bit Generator (PRBG) is a deterministic algorithm that given a truly random binary sequence. The input to the PRBG is called the seed, while the output of the PRBG is called a pseudorandom bit sequence of bits.

It is important to mention that the frequency, run and autocorrelation test are called the Main Binary Standard Randomness Tests.

A good Pseudo Random Number Generator should satisfies a set of statistical requirements, The statistical properties: output symbols should uniformly distributed; in binary symbol must be balance. There are variety of statistical tests, such as the frequency test, serial test, runs test, poker test, and autocorrelation test which considered as five stander tests [12].

1. Frequency Test

This test use to determine wither the number of 0's and 1's in a sequence *S* (key stream sequence) with length n are:

$$T_1 = \sum_{i=0}^{1} \frac{(n_i - n/2)^2}{n/2} = \frac{(n_0 - n_1)^2}{n} \quad \ldots (1)$$

Where n_0, n_1 denoted the observed number of 0's and 1's in S respectively. $n/2$ Is the expected value, n is the length of S which approximate follow a chi-square distribution with 1 freedom degree.

2. Serial Test

This test aims to determine whether that the number of occurrences of $00, 01, 10$ and 11 as a subsequence **S** are approximately the same as would be expected for random sequence

$$T_2 = \sum_{i=0}^{1} \sum_{j=0}^{1} \frac{(n_{ij} - E)^2}{E} \quad \ldots (2)$$

Where n_{00}, n_{01}, n_{10} and n_{11} denoted the observed number of $00, 01, 10$ and 11 respectively the expected value $E = (n-1)/4$,Where $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$.

$T_2 \sim \chi^2 (0.05, 3)$.

3. Poker Test

This test divide the sequence *S* in to k. parts with length $m \geq 3$ and let n_i be the observed number of occurrence of the i^{th} kind of sequence of length m. The poker test aims to determine whether the sequence of length m each appear approximately the same number of times in *S* as would be expected for the random sequence.

$$T_3 = \sum_{i=0}^{m} \frac{(n_i - E_i)^2}{E_i} \quad \ldots (3)$$

Where $E_i = C_i^m (1/2^m)\ (n/m)$, $T_3 \approx \chi^2 (0.05, m)$.

4. Run Test

A run of sequence S is a subsequence of S consisting of consecutive 0's or 1's, run of 0's called Gap while run of 1's called Block. Run test use to determine whether the number of runs of various lengths in the sequence S is as expected for a random sequence

$$T_4 = \sum_{i=1}^{k} \frac{(G_i - E_i)^2}{E_i} + \frac{(B_i - E_i)^2}{E_i} \quad \ldots (4)$$

Where $\boldsymbol{k}$ equal to the largest gap or block, B_i, G_i be the observed number of blocks and gaps respectively of length i in S for each i; $1 \leq i \leq k$. The expected value equal to $E_i = n - i + 3/2^{i+2}$, $T_4 \sim \chi^2 (0.05, 2k-2)$.

5. Autocorrelation Test

This test is use to check for correlations between the sequence *S* and (non-cyclic) shifted reversions of it. Let τ (number of shifting) be a fixed integer $1 \leq \tau \leq n/2$.

$$T_5 = (n_0(\tau) - n_1(\tau))^2 / (n - \tau) \quad \ldots (5)$$

Where $n_0(\tau)$ and $n_1(\tau)$ denote the observed of 0's and 1's in shifted sequence respectively, $T_5 \sim \chi^2 (0.05, 1)$.

3.2 Linear Complexity [1]

The linear complexity of a finite binary sequence S^n is the length of the shortest LFSR that generates a sequence having S^n as its first terms and denoted by $LC(S^n)$ and can by calculated by using Berlekamp-Massey Algorithm [13].

3.3 The Periodicity [14]

Let the P (S_i) represent the period of sequence S produce from the LFSR system, and let P(S_i) represent the period of each sequence produces from $LFSR_i$ for each $1 \leq i \leq N$. if $P(S_i) = (2^{r_i} - 1)$; r_i is the lengths of $LFSR_i$, so the periodicity equal to:

$P(S_i) = l.c.m\,(2^{r_i} - 1)\ ; i = 1 \ldots N$ … (6)

3.4 Correlation Immunity [15]

It's a relation between the output sequence of Combining Function Unit(CF) $CF = F_N$ from the KG and the sequences that are combined with each other by CF. This relation caused because of the high nonlinearity of the combined function F_N .The correlation probability (CP) of x in general, represent the ratio between the numbers of similar binaries (S_B) of two sequence to the length n of the compared part of them.

$CP = S_B / n$ … (7)

The best value of the correlation immune for any system when $m = N$ (m is the numbers of immune LFSRs); that's mean all x_i ,$1 \leq i \leq n$ Are statically independent from the output Z.

4. Design of New Dynamic Stream Cipher

The basic idea behind the stream cipher cryptosystem is generating key stream based on the use collection of LFSR's with fixed arrangement, all these LFSR's are filled depending on the value of the mixed initial key.

In this paper we will design a system based on a dynamic technique called the robust dynamic stream cipher cryptosystem **(RDSCC)**. This cryptosystem is dynamic because it has different structure of the combined LFSR's after each change in the seed key to get unanalyzable encryption algorithm.

This is done by the use of a bank of LFSR's stored in protected file and then selecting a random collection of LFSR's that are used in the algorithm to generate the encryption (decryption) key.

4.1 Key Management of RDSCC

There are two types of keys that are designed as an initial key for the LFSRs of the system, these keys are:

(1). **Message Key (MK)**: This key is non-secret which consist of (10) **ASCII CODE (8 bits)** characters. A new MK can be used with every new message to guarantee that no two messages have the same initial key. Before the encryption starts, the MK key is generated randomly by **RDSCC**. This key will be send with an encrypted message.

(2). **Basic Key (BK):** This key contains of high secret (20) **ASCII CODE (8bits)** characters and is changed daily. This key must be save in a protected file in both ends.

4.2 Basic Components of RDSCC

The proposed **RDSCC** includes the following main components:

A. Choosing Shift Register (CSR): It's a single LFSR with length 17.

B. Address Shift Register (ASR): It's a single LFSR with length 8.

C. Initialization System (IS): It consists of 4 fixed LFSR's with different length (37, 43, 31 and 53) and it's filled by the seed key (mixed of MK and BK).

D. Moving System (MS): Consists of 8 LFSR's and its changed every message.

E. Balance System (BS): Consists of 4 LFSR's and its changed every message.

F. Virtual Memory Unit (VMU): Consists of 256 random and different bytes.

4.3 Initialization of the RDSCC

A. First, we have to obtain a string of 160 bits called Bits of Key (BKE), this is done by the following steps:

I. Bits of BK: BBK_{ij}, i=1,…,20; j=0,…,7; k=(i-1)*8+j:

I	1	1	…	1	2	2	…	2	…	20	20	…	20
J	0	1	…	7	0	1	…	7	…	0	1	…	7
K	1	2	…	8	9	10	…	16	…	153	154	…	160
BBK	BK 1,0	BK 1,1	…	BK 1,7	BK 2,0	BK 2,1	…	BK 2,8	…	BK 20,0	BK 20,1	…	BK 20,7

II. Bits of MK: BMK_{ij}, i=1,…,10,1,…,10; j=0,…,7; k=(i-1)*8+j:

i	1	…	1	…	10	…	10	1	…	1	…	10	…	10
j	0	…	7	…	0	…	7	0	…	7	…	0	…	7
k	1	…	8	…	73	…	80	1	…	8	…	73	…	80
BMK	MK 1,0	…	MK 1,7	…	MK 10,0	…	MK 10,7	MK 1,0	…	MK 1,7	…	MK 10,0	…	BK 10,7

III. BKE_k, k=1,…,160: Also $BKE_k=BBK_{ij}$ XOR BMK_{ij}

k	1	…	8	…	73	…	80	81	…	88	…	153	…	160
BBK	BK 1,0	…	BK 1,7	…	BK 10,0	…	BK 10,7	BK 11,0	…	BK 11,7	…	BK 20,0	…	BK 20,7
BMK	MK 1,0	…	MK 1,7	…	MK 10,0	…	BK 10,7	MK 1,0	…	MK 1,7	…	MK 10,0	…	BK 10,7
BKE	KE 1	…	KE 8	…	KE 73	…	KE 80	KE 81	…	KE 88	…	KE 153	…	KE 160

B. The IS filled by the string BKE_k, k=1,...,160, then the last stages of each LFSR filled by 1.
C. From the BKE_k, k=1,..,160, choose 20 bits depending on the relation of k mod 8=0, the first 4 bits are combined with each other to choose one connection function for the ASR, CSR filled from the rest of 20 bits (16 bits). Then the last stage is filled by 1.
D. The CSR moves to generate 12 numbers ranged from 1-16 randomly that represent the indices of the LFSR's kept in LFSR's bank (with connection functions). The first 8 LFSR are specialized for MS and the other 4 LFSR's are for the BS.
E. The two systems MS and BS are filled by IS, also their last stages are filled by 1.
F. The IS moves again to generate 256 distinct bytes to fill the VMU.
See Figure -2.

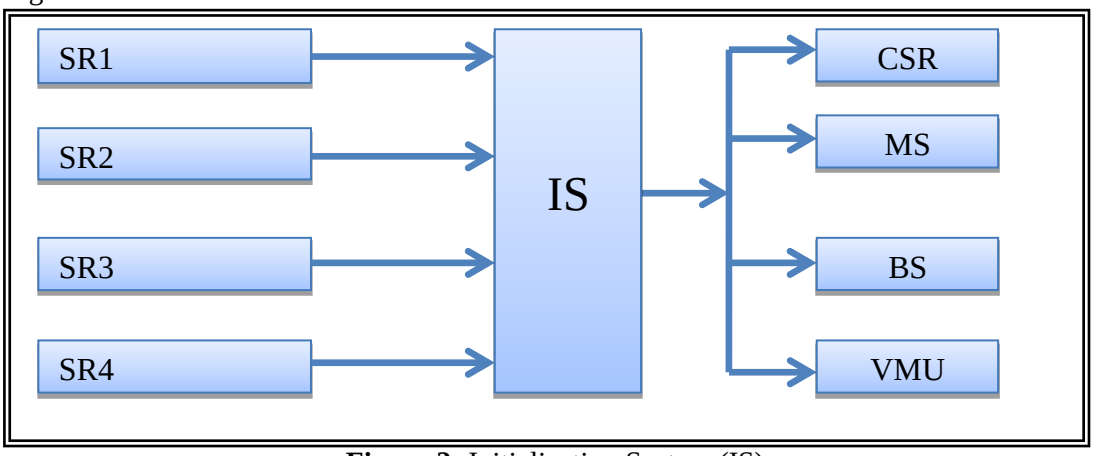

Figure 2- Initialization System (IS).

4.4 Moving of RDSCC

The proposed cryptosystem movement is as follows:

1. The MS moves to obtain address $AD=\Sigma x_i*256/2^i$, to VMU, where x_i is the output bit from $LFSR_i$ of MS, i=1,…,8.
2. $Byte_1=VMU(AD)$.
3. BS move to obtain $y_1,p_1,y_2,p_2,y_3,p_3,y_4,p_4$, where y_i is the output bit from $LFSR_i$ of BS, i=1,…,4, and p_i is a fixed position from $LFSR_i$ of BS, i=1,…,4.
4. These 8 bits are filled in ASR, this ASR moves to obtain address $NS=\Sigma b_i*16/2^i$, i=1,…,4, where b_1, b_2, b_3 and b_4 are bits obtained from fixed positions in LFSR's 1,3,5 and 7 from MS.
5. Let $Byte_2$ be the final byte of contents of ASR, s.t. $Byte_2=\Sigma c_i*256/2^i$, where c_i is the final bits of ASR.

6. The final key is key=$Byte_1$ XOR $Byte_2$.
   See Figure-3.

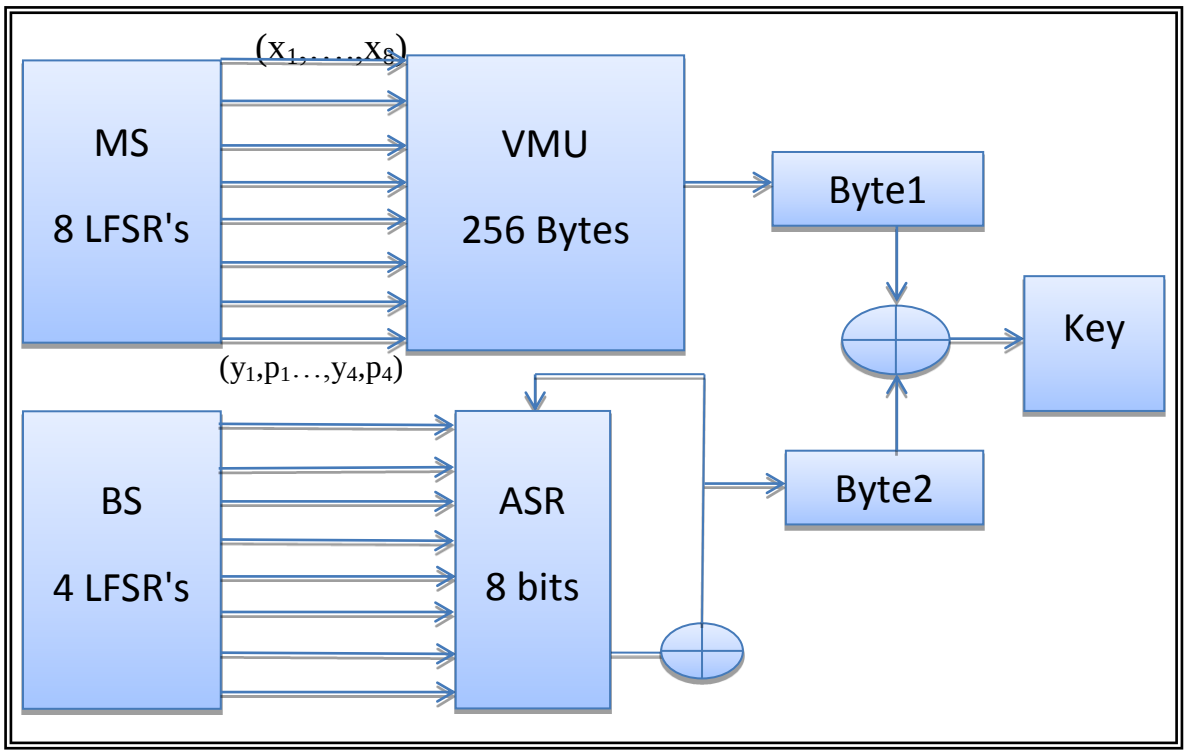

Figure 3-Moving of RDSCC.

RDSCC Algorithm

Step (1): input BK (20 chrs), MK (10 chrs), text_input (plain/cipher), with length L bytes;
Step (2): mixing BK and MK to initialize IS.
Step (3): Fill CSR system by IS to specify the shift registers of MS and BS from LFSRs bank.
Step (4): Move IS to fill MS, BS, and VMU.
Step (5): For i=1:L
AD=MS(x_j); j=1,..,8.
Byte1=VMU(AD);
ASR=BS(y_j,p_j); j=1,..,4.
Byte2=Move(ASR);
Key(i)=Byte1⊕Byte2;
Text_output(i)=Key(i)⊕text_input(i);
End {for i}
Step (6): Output text (cipher/plain);
End

The block diagram of RDSCC algorithm is described in Figure-4.

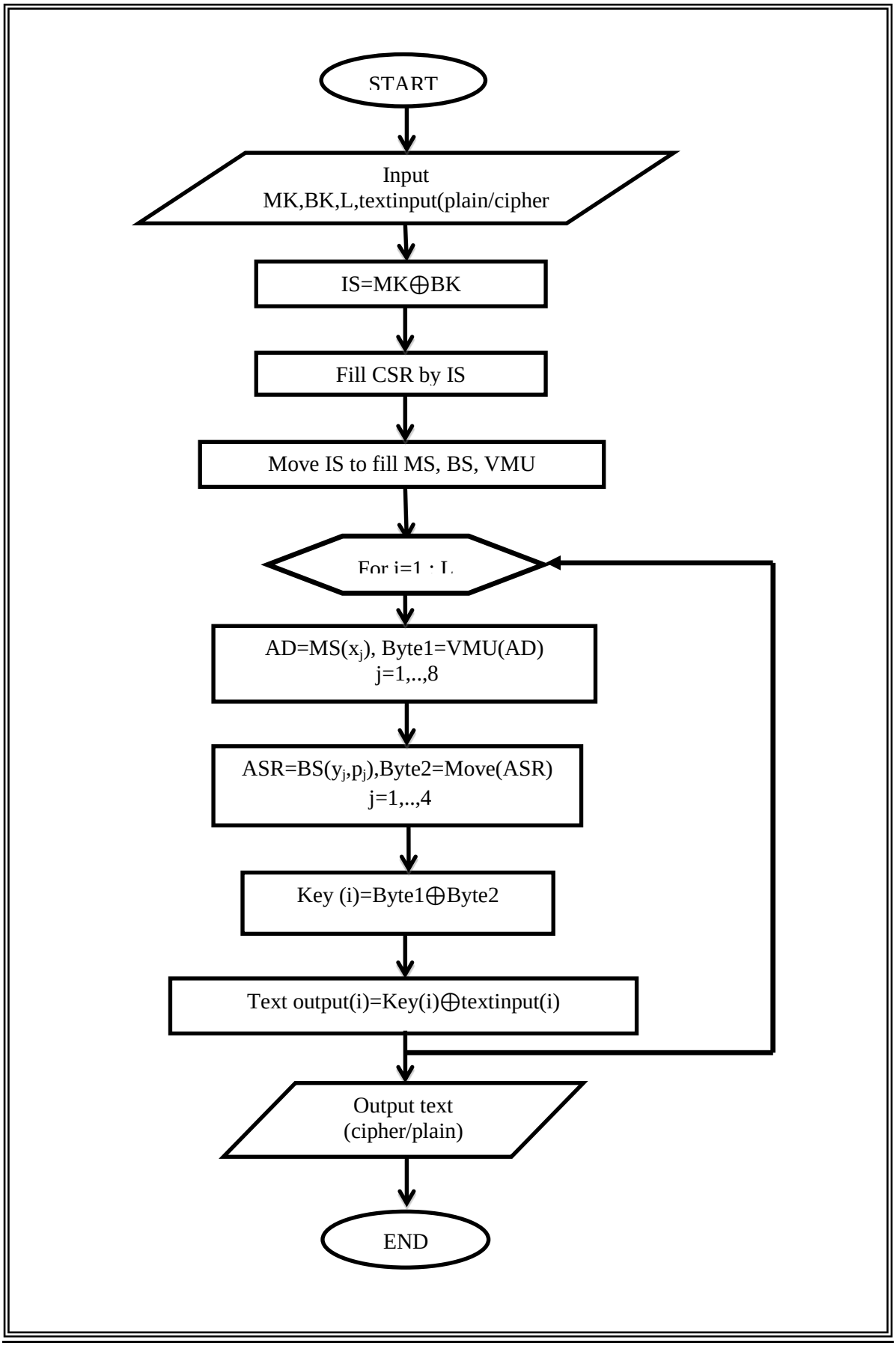

Figure 4-RDSCC Block diagram.

5. Implementation of BEC on RDSCC

We will review the results of output keys from **RDSCC** for various examples with different lengths (L_i, i=1,2,3), with different key management as follows:

1. Example1: L_1=1000 bytes (=8000 bits).
2. Example2: L_2=2000 bytes (=16000 bits).
3. Example3: L_3=5000 bytes (=40000 bits).

5.1 Periodicity

Table 1-Periodicity tests for three examples.

Test	Example1	Example2	Example3
Periodicity	2^{502}	2^{412}	2^{546}

5.2 Linear Complexity

Table 2-Linear complexity tests for three examples using Berlekamp-Massey Algorithm.

Test	Example1	Example2	Example3
Linear complexity	4000	8000	20000

5.3 Correlation Immunity

Table 3-Correlation Immunity tests for three examples.

Test	**CP for Key Length in Bits**			
Correlation Immunity	Example1	Example2	Example3	**CI acceptance**
MS1	49.90	51.45	50.54	**accept**
MS2	50.50	49.20	49.76	**accept**
MS3	52.90	48.80	49.36	**accept**
MS4	50.60	50.20	49.00	**accept**
MS5	51.90	50.45	48.66	**accept**
MS6	53.50	49.15	48.34	**accept**
MS7	49.80	49.85	50.34	**accept**
MS8	48.80	51.00	50.90	**accept**

Where MSj is the output of LFSR j in MS.

CI (**RDSCC**) =8 which equal the number of shift registers in MS.

5.4 Randomness Tests

In this section we will apply chi-square test on the results gotten from calculations of randomness tests.

Let k be the number of categories in the sequence S, c_i be the category i, $N(c_i)$ be the observed frequency of the category c_i, P_i the probability of occurs of the category c_i, then the expected frequency E_i of the category c_i is $E_i=P(S)\cdot P_i$, the T (chi-square value) can be calculated as follows:

$$T=\sum_{i=1}^{k}\frac{(N(c_i)-E_i)^2}{E_i} \quad \ldots(8)$$

Assuming that T distributed according to chi-square distribution by υ=k-1 freedom degree by α as significance level (as usual α=0.05%), which it has T_0 as a pass mark. If $T\leq T_0$ then the hypothesis accepted and the sequence pass the test, else we reject the hypothesis and the sequence fails to pass the test, this mean that T not distributed according to chi-square distribution (for more information about chi-square see [16] or any book in statistics and probability).Tables-(4, 5, 6, 7, 8) show the results of applying Frequency, Serial, Poker, Run and Autocorrelation tests.

Table 4-Results of applying Frequency test for three examples.

Test	Example1	Example2	Example3	υ	T_0	**Decision**
Frequency	0.0125	0.0063	1.8769	1	3.841	**pass**

Table 5- Results of applying Serial test for three examples.

Test	Example1	Example2	Example3	**υ**	**T_0**	**Decision**
Serial	4.0359	0.2582	4.5456	3	7.814	**pass**

Table 6-Results of applying Poker test for three examples.

Test	Example1	Example2	Example3	**υ**	**T_0**	**Decision**
Poker	2.9320	2.3800	5.0332	5	11.107	**pass**

Table 7-Results of applying Run test for three examples.

Run Test	**Max Run**	**T**	**υ**	**T_0**	**Decision**
Example1	13	23.5721	24.996	36.1306	**pass**
Example2	12	18.1194	21.026	33.6400	**pass**
Example3	17	44.5960	27.587	45.9098	**pass**

Table 8-Results of applying autocorrelation test for three examples.

τ	Example1	**Decision**	Example2	**Decision**	Example3	**Decision**
5	0.5615	**pass**	1.0404	**pass**	0.0110	**pass**
10	3.2040	**pass**	0.0023	**pass**	0.0049	**pass**
15	0.3007	**pass**	0.1502	**pass**	1.0104	**pass**
20	1.1549	**pass**	0.6258	**pass**	1.3232	**pass**
25	10.1850	**fail**	0.1383	**pass**	0.6324	**pass**
30	0.6504	**pass**	0.4008	**pass**	0.0036	**pass**
35	0.0554	**pass**	0.0527	**pass**	0.0210	**pass**
40	0.0020	**pass**	0.4634	**pass**	0.5046	**pass**
45	0.5985	**pass**	0.0076	**pass**	0.0421	**pass**
50	0.4528	**pass**	0.9954	**pass**	0.4101	**pass**

6. Conclusion

1. Since the proposed cryptosystem is dynamic, that's mean the MS and BS have different lengths of LFSR's with different order and that is obvious form Table-1.
2. From Table-1, we can conclude the high periodicity of RDSCC.
3. The RDSCC has high linear complexity and that proved in table (2).
4. Table-3 shows the good correlation immunity of RDSCC.
5. The RDSCC has good statistical randomness tests results that mean it has a good security and it can be used as encryption system (see Tables (4-8)).
6. The proposed cryptosystem can be used to encrypt not only texts, it may be used to encrypt images, videos, or any media files, because of its high security and speed.

References

1. Rueppel R. A. **1986.** "*Analysis and Design of Stream Ciphers",* Springer-Verlag, Berlin.

2. Vernam G. S. "Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications", *Journal of the American Institute of Electrical Engineers*, **55**.

3. Golomb, S. W. **1967.** "*Shift Register Sequences*" San Francisco: Holden Day 1967,(Reprinted by Aegean Park Press in 1982).

4. Smart, N. **2003.** "*Cryptography: An Introduction*", McGraw-Hill Education, 2003. ISBN 0-077-09987-7.

5. Briceno, M., Goldberg, I., and Wagner, D. **1999.** "*A Pedagogical Implementation of A5/1*", Available at http://jya.com/a51-pi.htm (accessed August 18, 2003).

6. Halevi, S., Coppersmith, D., and Jutla, C. S. **2002.** "*Scream: A Software Efficient Stream Cipher*", In J. Daemen and V. Rijmen, editors, Fast Software Encryption 2002, volume 2365 of Lecture Notes in Computer Science, pages 195–209. Springer-Verlag.

7. Johnson, D. W. and Johnson, F. P. **2002.** "*Joining Together: Group Theory and Group Skills*", Allyn & Bacon, July.

8. Zoltak, B. **2004.** "*VMPC One-Way Function and Stream Cipher*", In B. Roy and W. Meier, editors, Fast Software Encryption 2004, volume 3017 of Lecture Notes in Computer Science, pages 210–225. Springer-Verlag.

9. Chen, K. and et al, **2005.** "*Dragon: A Fast Word Based Stream Cipher*", eSTREAM, ECRYPT Stream Cipher Project, Report 2005/006 (2005-04-29).

10. Selmer, E. S. **1996.** "*Linear Recurrence Over Finite Field*", University of Bergen, Norway.

11. Beker, H., and Piper, F. **1982.** "*Cipher Systems: The Protection of Communications*", John Wiley & Sons, New York.

12. Stefanek, A. **2008.** "*M3P14 Elementary Number Theory*", Mathematics Imperial College, London.

13. Massey, J. L. **1969.** "Shift-Register Synthesis and BCH Decoding", *IEEE Transactions on Information Theory*, **15**: 122–127, 1969.

14. Menzes, A., Van Oorschot, P., and Vanstone, S. **1996.** "*Hand Book of Applied Cryptography*", CRC Press.

15. Clark, A., Golic, J. and Dawson, E. **1996.** "*A Comparison of Fast Correlation Attack*", D. Gollmann, editor, Fast Software Encryption, third International Workshop (LNCS1039), 145-157, Springer-Verlag.

16. Martinez, W. L. and Martinez, A. R. **2002.** "*Computational Statistics Handbook with MATLAB*", Chapman & Hall/CRC, Library of Congress Cataloging-in-Publication Data.