Design of Cryptographically Strong Generator By Transforming Linearly Generated Sequences

Matthew N. Anyanwu manyanwu @memphis.edu
Department of Computer Science
The University of Memphis
Memphis, TN 38152, U.S.A.

Lih-Yuan Deng lihdeng @memphis.edu
Department of Computer Science
The University of Memphis
Memphis, TN 38152, U.S.A.

Dipankar Dasgupta ddasgupt @memphis.edu
Department of Computer Science
The University of Memphis
Memphis, TN 38152, U.S.A

Abstract

Random numbers have been used extensively in many simulation applications like Monte Carlo Integration or computer modeling. But recently security applications have increased the need for strong (secure) random number generation like automatic password generation, encryption algorithms, on-line gambling etc. Thus random number generation has become a challengingand an interesting task. Most classical random number generators, generate sequences that are either linear or predictable hence not suitable for cryptographic and security applications. Others generate sequences that even though they are secure they are not cryptographicallystrong and above all are slow in execution. Also recent advances in random number generation like the construction of Multiple Recursive Generator(MRG) with large orders, Fast Multiple Recursive Gener-ator (FMRG) and DX(system of multiple recursive generators proposed by Deng and Xu [2003]) generators does not generate a strong random number sequences. Though MRGs have extremely long period of length with good empirical performance, its recurrence equation can be solved given a small set of its generated sequence, this implies that MRGs and FMRGs are not strong cryptographic generators. We propose an algorithm that will transform linear sequences generated by both classical LCG, MRGs, FMRGs and DX generators and make them cryptographically strong generators by hiding the entire sequence generated by the generators, thus it will be difficult for cryptanalyst to predict or infer the generator sequence if even the partial sequence or the parameters or knowledge of the algorithm used in the transformation of the generators
are known.

Keywords: Linear Congruential Generator (LCG), Multiple Recursive Generator (MRG), Random Number generation, Strong (Secure) generators and classical generator

1. INTRODUCTION

The quality of any simulation application or study depends to a large extent the quality of random number generator. In the same manner the security of any cryptographic application that uses random numbers depends on the strong nature of the random number generators. Classical generators like LCGs generates random sequences that are linear, fast, simple and easy to implement, thus they have wide range of uses in simulation applications. But recently there have been great demand for random generators by security applications like automatic password generation, on-line gambling, digital signatures, encryption algorithms etc. A random generator is cryptographically strong if it is not possible to predict the value of the next random bit and also if it is generally incompressible (See, Bruce [1994]). The efficiency of a good random number generator for cryptographic purposes is determined by the number of mistakes in predicting the next random bit and also if the prediction can be computed in polynomial time (See, Hugo [1998]). Cryptographic generators are also expected to be fast in generating its output sequences, and the sequences it generates should be difficult to predict or infer. There have been recent advances in random number generation to generate sequences that are cryptographically strong but experimental results have shown that these generators are still insecure as their next bits can be predicated. Recent advances in random number generation include MRGs with large order (See, Deng [2005]), FMRGs (See, Deng and Lin [2000]) etc. In Section 2, we review some of the classical generators and state why they are not cryptographically strong generators by enumerating their shortcomings. In Section 3, we review the recent advances in random generation and attempt to make these generators strong cryptographic generators. We also stated how to transform these generators into strong generators. In Section 4, we proposed an algorithm that will transform the sequences generated by classical linear generators like LCGs and recent generators to produce a strong cryptographic generator that will be difficult for a cryptanalyst to predict or infer. Our algorithm transform the random bits generated by both linear classical generators and MRGs into a cryptographically strong bits by performing a transformation on the original generated bits and then perform a bitwise exclusive-or operation between the original bits and the transformed bits. The transformation of the bits is used in hiding the generated bit sequence from the attacker. The transformation also breaks the linear structure of the linear sequences making them difficult to predict. The final output are sequences that are difficult to predict and strong for cryptographic purpose. In Section 5, we did a comparison between sequences generated by classical generators and our transformed sequences.

2. Classical Random Number Generators

Pseudo-random number generators (PRNG) are random number generators that produce sequence of values based on an initial seed and current state. They are called deterministic generators in that given the same initial seed they will output the same sequence values. Random numbers have been applied in simulation experiments, randomized algorithms, and Monte Carlos methods in numerical analysis and also in cryptography (See, Neal [1994]). PRNGs used for cryptographic purposes are expected to be fast (computable in polynomial time) and secure (See, Stinson [2006]). But classical generators do not meet these two objectives simultaneously. Most classical generators like RSA and Blum-Blum-Shub (BBS) which are called power generators are cryptographically secure but too slow and cumbersome in performance for cryptographic applications. Other classical generators based on linear congruencies like Linear Congruential Generator (LCG) or linear feedback shift registers (LFSR) are very fast and easy to implement. But these linear generators are not cryptographically secure (the next bit stream value generated can be predicted), thus they are not good for cryptographic applications A PRNG is cryptographically secure if it is computationally infeasible to predict the value of the next random bit, even with knowledge of the algorithm, the hardware that generates the sequence and also the previous bit
streams (See, Bruce [1994]). Also secure PRNG are generally incompressible (See, Bruce [1994]), except when given the key (algorithm). In this section we will review some early classical

generators like RSA, BBS, BM (Blum Micali), LCG, LFSR and recent advances in pseudo-random number generators (Multiple Recursive Generators (MRG)

2.1 Linear Congruential Generator:

Linear Congruential Generator (LCG) was proposed by Lehmer [1951]. It uses a linear function over modulus arithmetic field. LCG is one of the most widely used PRNG, especially in simulation applications. It is produced iteratively as shown in the equation below;

$$Xi = (BXi-1 + c) \bmod m, \quad i \geq 1, \quad (1)$$

where Xi, B, c, and m are positive integers and are called the parameters of LCG (Xi are all integers between 0 and m-1). The quality of the generator depends on the selection of the increment c, multiplier B, initial seed X0, and modulus m. If c =0, the LCG is called mulplicative linear congruential generator (MLCG) as shown in the equation below;

$$Xi = (BXi-1) \bmod m, \quad i \geq 1, \qquad (2)$$

The number sequence generated by the LCG is of the form X0, X1, X2... , where X0 is the initial seed. A series of researches and experiments have shown that LCG sequences can be easily predicted and this raises some doubt about the security of LCG when it is used in any cryptosystem, thus it is not suitable for cryptographic applications.Boyar [1982] and Bruce [1994] showed that if the whole sequence of LGC is published, the sequence can be predicted if the initial seed is given, even though other parameters (B, c or m) of LCG may be unknown. LCGs have been broken as specified in articles (See, Boyar [1989] and Reeds [1977]). Thus LCGs are not good cryptographic generator but very useful in non-cryptographic applications like simulation. An on-line poker game that uses random numbers generated by LCG to shuffle deck of cards will be flawed because the LCG sequences can be easily be predicted and this will lead to cheating in the game if the sequences generated by LCG are not made secured.

2.1.1 Predicting LCG Generator

Boyar [1982] used the Plumstead's algorithm to predict the LCG sequence using partial consecutive sequence of the generator, the number of the consecutive sequence used depends of the size of the modulus. In simulation of the Plumstead's algorithm by Pommerening (See, Pommerening [2003]), when the size of the modulus is of the order of (231 – 1), ten consecutive sequence of the generator are needed for the prediction of LCG sequence. There are also many other methods of predicting the sequence of LCG which depends on LCG parameters given. If the multiplier (B) is unknown but the partial sequence and the modulus are known then the LCG (2) can be predicted by calculating the modulus inverse as stated below; B=X1*X0-1 mod m,Where X1, X0 are the consecutive variates of the LCG. The value of the multiplier B will then be substituted to predict the LCG sequence (2). If only the partial sequence of LCG generator is given, the modulus, multiplier and the seed of the LCG can be determined using Haldir method of cracking LCG generator (See, Haldir [2004]). Haldir cracking of LCG generator is based on George Marsaglia analysis of pseudorandom random number generators. Marsaglia pointed out that there is a flaw in pseudorandom generators as their uniform variants when viewed as points in a cube fall into hyper-plane, which contains points that are well distributed in a unit cube. The LCG sequence is used in forming a matrix, the determinant of the matrix will give an integer multiple of the modulus m. Then the gcd of a number of matrices gives the actual value of the matrix. Given the LCG as in (1) where c =0, if X0, X1, X2... are observed from the sequence of LCG, then they can be used to
form a matrix thus;.

$$\begin{matrix} X1 & X2 \\ X2 & X3 \end{matrix}$$

The determinant $D = X1 \times X3 - X2 \times X2$. Note that

D mod m = 0(since D is an integer multiple of modulus m. Thus modulus m = GCD (D, for some number of matrices (about 4 or 5).
If the LCG is as in (1) where there is a constant, the matrix formed is as given below using Haldir cracking of LCG generator method as shown below;

X1 X2 1

X2 X3 1
X3 X4 1

Given the LCG sequence of consecutive numbers; 207482415, 1790989824, 2035175616, 77048696, 24794531,

109854999, 1644515420, 1256127050. Using Haldir method, the LCG sequence is used to form a 3 × 3 matrix of

the form;

207482415 1790989824

1790989824 2035175616
2035175616 77048696

1

1
1

The implementation of Haldir's method using the LCG sequence predicts the LCG sequence thus; m =2147483647, k= 16807 and seed =12345.
When we applied Haldir's method to our proposed algorithm(as in section 4), the transformed LCG sequence is not predicted, because the transformed LCG sequence are hidden by applying our proposed algorithm in section 4, making it computationally difficult for an attacker to infer the sequence of an transformed LCG sequence. Thus knowing the partial sequence or some of the parameters of the transformed LCG generator does not pose a threat any longer as our proposed algorithm is applied to the transformed LCG sequence.

2.2 Linear Feedback Shift Register (LFSR) Generator:

LFSR is used in generating sequence of binary bits. It is a PRNG whose input bit is a linear function of two or more bits. It uses a short seed to produce sequence of bits which has a very long cycle. LFSR can be said to be a special type of Multiple Recursive Generator (MRG) but it differs from MRG in that it generates binary sequence of bits while MRG generates sequence of numbers. It is based on the linear recurrence which is of the form stated below (See, Tausworthe [1965]);

$$Xi = (\alpha 1 Xi{-}1 + \bullet\bullet\bullet + \alpha k Xi{-}k) \bmod 2, \qquad (3)$$

The linear recurrence of LFSR is based on Z2 recurrence. The sequence of values produced by LFSR is determined by its current or previous state. The parameters of LFSR are k > 1 (the order

of the recurrence), Xi (bit generated), α (the multiplier) and X0 the initial seed. The maximum period of the recurrence is 2k – 1, if and only if the characteristics polynomial is as given below;

$$F(x) = 1 + \alpha 1x + \alpha 2x2 + \bullet\bullet\bullet + xk \quad (4)$$

The characteristic polynomial (4) is a primitive polynomial in Z2, which is a finite field with 2 elements (See, Lidl [1994]). The recurrence of (3) shows how each bit of the state evolves and also that each bit of the output is a linear combination in Z2 of the given state (See, Tausworthe [1965]). LFSR like LCG is a fast generator; it produces linear sequence of bits. LFSR sequence like that of MRG can be predicted as stated in section 3.1.1. Also LFSR bit-string sequence can be predicted using Berlekamp-Massey algorithm (See, Massey [1969] and Berlekamp [1968]).

2.3 Blum-Micali (BM) Generator

BM is a cryptographically secure generator presented by Blum and Micali [1982]. It is based on the difficulty in computing discrete logarithms [See, Stinson 2006], which is also based on the believe that the modular exponentiation modulo is prime and a one-way function. Thus it is based on the underlying discrete logarithm over a finite field and also on the assumption that solving discrete logarithm problems is a hard problem even when
the exponent is small.
The general form of BM is as given below;

$$Xi+1 = aXi \bmod m, \quad i \geq 0 \qquad (5)$$

The output of the generator is 1 if Xi < m/2, otherwise 0. The constant a is a primitive root in the finite field.
This generator is secure if the modulus m is very large (1024-bit modulus) thus it is computationally difficult to compute the discrete logarithms mod m. BM is not suitable for cryptographic applications even though it is secure, because the generation of the output sequence is very slow.

2.4 Blum Blum Shub(BBS) Generator:

Blum, Blum and Shub [1986] invented the BBS pseudo-random number generator. BBS is based on the hardnessof quadratic residues and inter factorization over modulus field (See, Stinson [2006], Raja and Hole [2007]). Itis a simple and secure cryptographic generator. BBS is generally of the form;

$$Xi+1 = (Xi)2 \bmod m, \quad i \geq 1 \qquad (6)$$

$$Zi = Xi \bmod 2 \qquad (7)$$

Zi is the output of the generator.

The modulus m of BBS is the blum integer. The relationship between the modulus m and the two large prime integers p and q is given below.
$m=p\times q$ and $p \equiv q \equiv 3 \bmod m$.

The security of this generator rests on the ability of the cryptanalyst to factor m. Also only one bit of the sequence is generated at a time instead of the whole number (See, Boyar [1989]), this makes the generator very slow in generating its sequences. Also it cannot be used as a strong cryptographic generator because it has been shown that in the Blum protocol of BBS, the modulo m can be factored during key exchanges (See, Hastad and
Shamir [1985]).

2.5 RSA Generator

RSA generator is based on the assumed security of RSA cryptosystem proposed by Rivest, Shamir, and Adleman[1978]. It is based on the hardness of integer factorization of the modulus number (See, Stinson [2006]). RSA forms a sequence of numbers in which each element in the sequence is an RSA encryption of the previous element. The least bit of the element forms the bit-string. The general form of RSA is as stated below;

$$X_{i+1} = X_i^e \bmod m, \quad i \geq 1 \qquad (8)$$

$$Z_i = X_i \bmod 2 \qquad (9)$$

Zi is the output of the generator.
The modulus m is a product of two large prime numbers p and q (512-bit primes), e is chosen so that

$$\gcd(e, \varphi(m)) = 1 \qquad (10)$$

where m and e are public key while p and q are the secret keys. The security of RSA is based on finding the factors of m, If m is large enough(1024-bit modulus) it will be difficult to factor it, RSA is a very slow generator, as only one bit of the sequence is generated at a time instead of the whole number.

3. Recent Advances in Random Number Generation

The classical generators like BBS, BM, RSA, LCG and LFSR described above are not good candidates for security applications. While BBS, BM and RSA are secure, they are slow in generating their bit sequences yet they are being used in some security applications as they are readily available. LCG and LFSR are not secure but they are fast in generating their sequences and also easy to implement. Cryptographically secure generators are supposed to be fast in generating its sequences and also its bit sequence is supposed to be difficult to predict

3.1 Multiple Recursive Generators (MRGs)

MRG is one the most extensively studied and most widely used PRNG which is based on the k-th order linear recurrence. The next value of MRG is computed recursively based on its previous values (See, Lehmer [1951]). The basic equation of MRG is thus stated below;

$$X_i = (\alpha_1 X_{i-1} + \bullet\bullet\bullet + \alpha_k X_{i-k}) \bmod m, \; i \geq k \qquad (11)$$

The starting values are (X0, X1, X2... Xk−1), which are not all zero; m is the modulus which is a large prime number and Xi can be transformed using Ui = Xi/m. In order not to obtain 0 or 1, (See, Deng and Xu [2003]) transformed
Ui = (Xi +0.5)/m. The parameters of MRG are Xi (generated sequence), αk(the multiplier), m(the modulus)and the order k. The maximum period of an MRG of an order k is mk − 1, which is obtained if the characteristic polynomial is as given below;

$$f(x) = x^k - \alpha_1 x^{k-1} - \bullet\bullet\bullet - \alpha_k, \qquad (12)$$

is a primitive polynomial and the modulus is prime. MRG is an improvement of LCG as it has good empirical performance, long periods and higher dimensional uniformity when compared to LCG. MRG reduces to LCG when its order k=1. When the order k of MRG is large, it becomes less efficient as it needs several multiplications compared to LCG which needs only one multiplication. To improve the efficiency of MRG many authors and researchers proposed improvements to it. Deng and Lin [2000] proposed FMRG which needs only one multiplication like LCG, thus in addition to features inherent in MRGs, FMRG is very fast, making it as efficient as LCG. Deng and Xu [2003] and Deng [2005] improved the performance of FMRG by proposing a system of DX generators which are fast, portable and efficient with maximal period compared with FMRG. They fixed the coefficient of the non-zero multipliers to be equal. The DX (DX-k-s(s =1 t4)) generators by
Deng and Xu [2003] are as stated below:

1. DX-k-1 [FMRG] ($\alpha_1 = 1, \alpha_k = B$).

$$X_i = X_{i-1} + BX_{i-k} \bmod m, \quad i \geq k. \quad (13)$$

2. DX-k-2 ($\alpha_1 = \alpha_k = B$).

$$X_i = B(X_{i-1} + X_{i-k}) \bmod m, \quad i \geq k. \quad (14)$$

3. DX-k-3 ($\alpha_1 = \alpha_{\lceil k/2 \rceil} = \alpha_k = B$).

$$X_i = B(X_{i-1} + X_{i-\lceil k/2 \rceil} + X_{i-k}) \bmod m, \quad i \geq k. \quad (15)$$

4. DX-k-4 ($\alpha_1 = \alpha_{k/3} = \alpha_{\lceil 2k/3 \rceil} = \alpha_k = B$).

$$X_i = B(X_{i-1} + X_{i-\lceil k/3} + X_{i-\lceil 2k/3 \rceil} + X_{i-k}) \bmod m, \quad i \geq k. \quad (16)$$

3.1.1 Predicting MRG Generator

The characteristics equation of MRG as defined in (12) has been determined by the first 2k terms of the sequence using a system of k equations (See, Lidl and Niederreiter [1994]). MRG sequence with the modulus and the partial sequence known can also be predicted, when the system of linear equations which can be formed using the recurrence of (11) is solved (See, Stinson [2006]) as shown below;

- $\alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_k X_k = X_{k+1} \bmod m$
- $\alpha_1 X_2 + \alpha_2 X_3 + \cdots + \alpha_k X_{k+1} = X_{k+2} \bmod m$
- .
  .
  .
- $\alpha_1 X_k + \alpha_2 X_{k+1} + \cdots + \alpha_k X_{2k-1} = X_{2k} \bmod m$

These systems of linear equations can be re-written in matrix format as

$$\begin{bmatrix} X_{k+1} \\ X_{k+2} \\ \vdots \\ X_{2k} \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & \cdots & X_k \\ X_2 & X_3 & \cdots & X_{k+1} \\ \vdots & & & \\ X_k & X_{k+1} & \cdots & X_{2k-1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{bmatrix}$$

The above equation can also be re-written as X (column), α (column) and M (matrix) as shown below,$X = M^* \alpha \pmod m$. (17)This gives rise to k equations in k unknowns. If the matrix M (k × k) in (17) has an inverse (mod m), then the solution to (17) can be obtained thus; $\alpha = M^{-1*} X \pmod m$. (18)The values of the multipliers (α1, α2, •••, αk) can then be obtained by solving (18). Hence this will lead to the prediction of MRG sequence of (11), given the modulus m. If the modulus of MRG is not known but the partial sequence (Xk+1,Xk+2,Xk+3,...,X2k) (at least 2k terms) for k-th order MRG is given, the sequence of the MRG can be predicted by forming a matrix of (k + 1) rows using consecutive terms of the sequence as done in the case of LCG. The determinant D of the matrix can then be calculatedcalculated is an integer multiple of the modulus m, thus D (mod m) =0. The actual modulus is calculated thus m = gcd (D, for a number of matrices (about 4 -5)). The sequence of the MRG is then used to form a matrix equation; the modulus is substituted and used in solving the equation of (18) to calculate other parameters of the MRG. Given the following MRG sequences; 1870519315, 618460120, 287572083, 2134648799, 83050304, 1713934911, 1872854851 with initial seeds of X0 = 12345678, X1 = 207482415, X2 = 79098924 and modulus m = 2147483647, of k-th order =3, we used (17) and (18) to obtain the parameters of the MRG and hence predict its sequence as
shown below;

$$\begin{bmatrix} 2134648799 \\ 83050304 \\ 1713934911 \end{bmatrix} \begin{bmatrix} 1870519315 \end{bmatrix}$$

=
618460120

287572083

618460120	287572083	α1
287572083	2134648799	α2
2134648799	83050304	α3

The solution to the above matrix equation in modulus m (we developed a c-code using NTL (Number Theory Library) packages to solve the matrix equation) gives the parameters (multipliers) of the MRG, the solution is as
stated below;
(α1, α2, α3) = (16807, 1036675, 1047849).
The MRG sequence can be predicted by substituting the parameters obtained in (11), which will give the matrix equation as

$$X_i = (16807X_{i-1} + 103667X_{i-2} + 1047849X_{i-3}) \bmod 2147483647, \; i \geq 3 \quad (19)$$

where X0 = 12345678, X1 = 207482415, X2 = 79098924. Using (19), we predicted the sequence of the MRG.
When this method of obtaining the parameters of MRG is appliedto our proposed algorithm (section 4) for the MRG, the parameters of the MRG was not obtained because our proposed algorithm hides the MRG sequence making it difficult for an attacker to infer or predict the sequence. Thus knowing the partial sequence or any of the parameters of the MRG does not pose a threat any longer when our proposed algorithm is applied.

4. Algorithm for secure and efficient pseudo-random number generator

In order to make LCGs and MRGs and other linear PRNG strong cryptographic generators. We propose an algorithm that will transform the linear sequence generated by the PRNGs. The transformation (20) hides the generated bits sequence, breaks the linear structure of the sequences and then perform a bitwise exclusive OR between the original bits and the transformed bits

The output of our algorithm gives a bit sequence which will be practically impossible for an attacker to infer in computational time. Our algorithm is stated below;

$$Y_i = X_i \; T(X_{i+1}). \quad (20)$$

In transforming the generated bits, we first truncate the bit performing a binary shift operation as shown
below;
STEP 1: A transformation is performed on the sequence (Xi)

$$X_i = (b_1, b_2 \ldots b_n) \bmod 2 \quad (21)$$

The transformation produces T (Xi+1) given below;

$$T(X_{i+1}) = (b_n, b_{n-1} \ldots b_2, b_1) \bmod 2. \quad (22)$$

STEP 2: A bitwise exclusive-or operation is then performed between the transformed and the original generated sequences as shown below;

$$Y_i = X_i \; T(X_{i+1}), \quad (23)$$

where denotes exclusive OR operation between the transformed sequence and the original generated sequence. The exclusive OR operation is used in producing a secure cryptosystem as explained in the next subsection. The final output of our algorithm (23) gives a cryptographically strong pseudorandom sequence which is computationally difficult to infer or predict. The transformed sequence hides the bits for each of the generated output sequence and breaks its linear structure making it impossible for the cryptanalyst to infer or predict the linear equence or solve the characteristics equation of the MRGs by a system of k equations Our algorithm not only hides the originally generated bits but it performs exclusive OR operation between the transformed bits and originalbits. The exclusive OR operation is necessary to ensure that the final output sequence is computationally
difficult to infer.

4.1 Simulation of the Proposed Algorithm

We applied our algorithm as stated in (23) to equations (2), (13), (14), (15) and (16). We implemented our algorithm using c-code supplements, various parameters of both LCG and MRGs were used. The algorithm is based on a 32-bit binary sequence. Our algorithm produces a secure generator that is very fast and computationally difficult to infer or predict, this is evidence by the following features;

• Our algorithm generates its bits sequence in polynomial computable time; the generation of the bits is also very fast.
• We used linear sequences (LCGs, MRGs) by Deng [2005, 2008] that passed standard statistical tests, thus both the input and output bit-strings of the algorithm are well distributed.
• The algorithm uses initial seed of short sequences and generates output long sequences bit-strings that are well distributed.
• Our algorithm hides the entire bits of the bit-string generated by linear generators, making it computationally difficult for an attacker to infer or predict the bit string of the generator.

Our Algorithm hides the bits generated by the linear generators using the following techniques;
• Our algorithm transforms the bit-string sequence by reversing the order of the bits of the bit-string after
truncating its lower bits before performing a Bitwise Exclusive-OR operation on the bits.
• The reversal of the bit-string sequence by our algorithm breaks the linear structure of the linear sequences
producing a non-linear sequence. Also non-linear sequence is more secure when compared with linear
structure as evidenced by BBS, BM and RSA generators
• The Bitwise Exclusive-OR between the transformed bit-string and the original bit-string by our algorithm
is used to toggle the positions of bits in the original bit-string with the transformed bit-string making it
computationally difficult for the attacker to predict correctly the position of the bits in the original bit-string.
• The Bitwise Exclusive-OR between the transformed bit-string and original bit-string is used to produce a
cipher bit-sting. Thus when an attacker attempts to predict the bit-string he will only succeed in getting
 the cipher bit and not the original bit. This makes the bit-string produced by our algorithm to be computationally difficult to predict
• The Bitwise Exclusive-OR operation introduces a piling-up lemma (See, Lemma 3.1 of Stinson [2006, page

81]), which ensures that the binary variables are independent, that is the state of one has no effect on the
state of any of the others which also ensures equi-distribution among the binary bits. Thus it is difficult for
an attacker to predict the transformed even if some part of the sequence is revealed.
Thus our algorithm produces a strong generator by transforming the linear sequences produced by LCG and MRGs into unpredictable sequence of bits.

4.2 Security of the Proposed Algorithm

The figures below shows that the sequence generated by linear generators is linear in structure (figure 1) while our transformed sequence is not linear (figure 2). We used LCG (B, m) to denote LCG (2) with multiplier B and prime modulus m. Thus we used LCG (B=3, m=127) to show that LCG sequence is linear (figure1), while the transformed LCG (B=3, m=127) is not linear. Figure 2 shows that our algorithm breaks the linear structure of the linear generators sequences, thus making them more difficult to infer or predict. The linear structure of the linear sequences is broken when our algorithm is applied by reversing the order of the bits, shifting the bits by one position to the right, and performing exclusive-OR operation between the original bit sequence and the transformed bit sequence. Also the same result in figure 1 and figure 2 will be obtained if MRGs and MRG ransformed by our algorithm is used. The security property of our proposed algorithm is based on the reversal of the bits sequence and the bitwise exclusive-OR operation between the original bit sequenceand the reversed bit sequence. Our proposed algorithm has transformed the linear sequence of LCG and MRG into non-linear sequence. Thus a hacker can obtain information about the parameters of LCG and MRG generators as stated above(for LCG generator) and given in the Table 1(for MRG generator)and attempt to used it to break our proposed algorithm but our algorithm hides the entire sequence generated by the linear generators making it difficult for him to crack the algorithm, hence knowing some of the partial sequence or the parameters of the linear generators LCG(B, c and m) and MRG(K, αk, and m) will no longer pose a security threat to the generators, as they cannot be used to solve for or predict the sequence of the generators. This makes our algorithm difficult to infer by attackers. Also if the hacker has information about the algorithm we used in transforming the generator sequences, it will be difficult for him to reverse (invert) the sequences back to linear sequence. Thus the transformed sequence now behaves like a non-linear sequence, which is difficult to infer or break.

4.3 Recommended Parameters for Our Proposed Generator

The modulus of the generators (LCG and MRG) transformed by our proposed algorithm and the one we have been using for our experiments is rather too small to be used for practical purposes. Park and Miller [1988] published minimal standard for LCG parameters. The minimal standards are large prime numbers of modulus m =231 – 1, B=16807 and c (constant term) =0 for 32-bit CPUs. We recommend this minimal standard for LCG parameters when our proposed algorithm is applied. Table1 below shows some of the parameters of DX generators by Deng [2008] determined by efficient search algorithm. We recommend the parameters in table 1 for our
proposed algorithm used in transforming the sequence of MRG (DX) generators. The parameters in table 1 are used in producing efficient and portable multiple recursive generators of large order with good empirical performance and a period of approximately 1093384 (See, Deng [2008]).

Fig.1: X(i+1)/m vs X(i)/m for LCG(B=3,m=127)

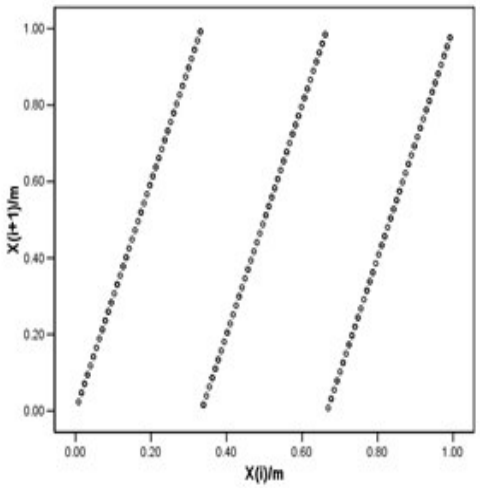

Fig.2: Y(i+1)/m vs Y(i)/m for the trasnformed LCG(B=3,m=127)

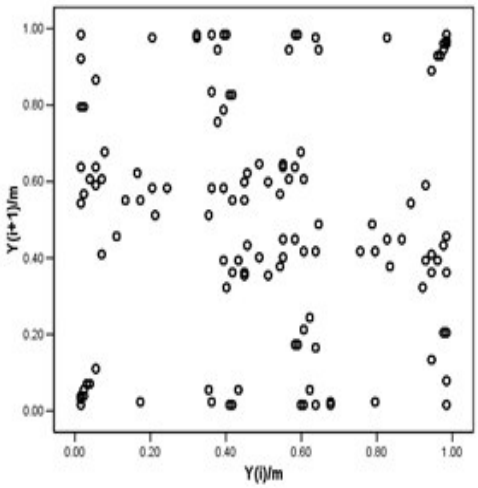

Table 1: List of k, m and $\alpha_c 2^{30}$ for DX-k-s generators

k	m	s=1	s=2	s=3	s=4			
5003	2146224359	1073727083	1073741516	1073730698	1073740466			
6007	2137498943	1073738651	1073715261	1073729141	1073738504			
7001	2146873559	1073709808	1073728419	1073738188	1073735327			
8009	2142326903	1073717208	1073726014	1073733016	1073719175			
9001	2140247399	1073737583	1073717540	1073733156	1073732451			
10007	2147051903	1073726195	1073702542	1073723329	1073730725			

5. Comparison between Classical Generators and Transformed Generators

As stated in proceeding sections classical generators, that are fast in generating their output sequences like LCGs are not cryptographically strong since their sequences can be predicted. Also classical generators that are secure are not suitable for cryptographic applications as they are very slow in generating their output sequences since the generators produces one bit at a time rather than the entire sequence at each step of the generation (See, Boyar [1989]) but they are still been used in security applications they are readily available. Classical generators like LCG can be easily predicted if the previous elements of the sequence and the initial value are given, without knowing the modulus and the multiplier (See, Hugo [1999]). LCG can also be predicted if the modulus and the previous sequence is given even though the multiplier and the initial seed are not known by using the inverse of the modulus (See, Frieze and Langarias [1984]). MRGs, FMRGs including DX generators can be predicted if the characteristics polynomial equation can be solved using a system of k equations (See, Lidl and Niederreiter [1994]). The BBS, BM and RSA generators are secure but they are not very fast in generating their bits sequence, the output sequence is generated one bit at a time . Most classical generators are linear generators in that the structure of the generated sequences is too regular

and have a coarse lattice structure (See, Takashi et al [1996]). But the transformed sequence by our algorithm is fast, efficient, non-linear and strong, also difficult to predict or infer.

6. Conclusion

Experimental results and tests have shown that classical generators like LCGs that generate pseudorandom linear sequences are not suitable for cryptographic purposes, even though it is simple, efficient and easy to generate. Other classical generators like BBS, RSA, and BM etc that are thought to be secure are equally not good enough for cryptographic purposes as they are slow in generating the next random bit sequence. Also the recent advances in random number generation (MRGs and FMRGs) are fast and efficient in generating linear sequences with long periods and good empirical performance, but still they are not cryptographically strong as the linear system can be predicated using a system of unique k equations. Our proposed algorithm produces a strong pseudorandom sequence that is suitable for cryptographic purposes and difficult to predict/infer by transforming the linear sequences and breaking its linear structure. The transformation hides the linear bits of the generated linear sequence preventing the attacker from accessing the generated output sequence, even with the knowledge of the partial sequence, parameters of the generators and the algorithm used in transforming the generator sequence. Thus knowing the parameters and partial sequence of the generators does not pose any threat any longer as the prediction of the generator sequence will no longer be an easy one. Also the knowledge of the transformation algorithm will not be of much use to the attacker as he will find it difficult to reverse (invert) the transformed sequence back to linear sequence. The bits sequence generated by our algorithm has features described in section 4.2, which makes the output sequence to be fast, strong and also difficult to predict or infer.

References:

1. Alexi, W., Chor, B. Z.,Goldreich, O. and Schnorr, C.P. (1988). RSA and Rabin Functions: Certain Parts Are as Hard as the Whole Proceedings of the 25th IEEE Symposium on the Foundations of Computer
Science 449-457.

2.Berlekamp, E.R. 1968. Algebraic coding theory McGraw-Hill, New York, (1968).

3.Blum, M., and Micali, S. (1982). How to generate cryptographically strong sequences of pseudo-randombits.In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science. IEEE, New York.
pp. 112- l 17

4.Blum, L., Blum, M., and Shub, M. (1983). A simple secure pseudo-random number generator. InAdvances in Cryptography: Proceedings of CRYPTO 82. Plenum Press, New York, 61-78.

5.Blum, L., Blum, M., and Shub, M. (1986). A simple unpredictable pseudo-random number generator.
SIAM Journal on Computing, Volume 15, Issue 2, pp. 364-383 . ISSN:0097-5397.

6.Boyar, J. (1982). Inferring sequences generated by a linear congruence. Proceedings of the 23rd AnnualIEEE Symposium on the Foundations of Computer Science pages 153-159.

7.Boyar, J. (1989). Inferring sequences produced by pseudo-random number generators. Journal of theAssociation for Computing Machinery 36,129-141.

8.Bruce, S. (1994). Applied Cryptography: Protocols, Algorithms and Source Code in C John-Willey andSons, New York, 1994. ISBN: 0-471-5975602

9.Deng, L. Y., and Lin, D. K. J. (2000). Random number generation for the new century. AmericanStatistician 54, 145-150.

10. Deng, L. Y. and Xu, H. Q. (2003). A System of High-dimensional, Efficient, Long-cycle and Portable Uniform Random Number Generators, ACM Transactions on Modeling and Computer Simulation, Vol. 13,
No. 4, pages 299-309.
11. Deng, L. Y. (2004). Generalized Mersenne prime number and its application to random number generation, in Monte Carlo and Quasi-Monte Carlo Methods 2002 (H. Niederreiter, ed.), Springer-Verlag, 167-180.
12. Deng, L. Y. (2005). Efficient and Portable Multiple Recursive Generators of Large Order, ACM Transac-
tions on Modeling and Computer Simulation(TOMACS), Volume 15, Issue 1 PP.1-13. ISSN: 1049-3301
13. Deng, L. Y. (2008). Issues on Computer Search for a Large Order Multiple Recursive Generators, in Monte Carlo and Quasi-Monte Carlo and Quasi-Monte Cralo Methods 2006(S. Heinrich, A. Keller and H. Niederreiter, ed.) Springer-Verlag, 251-261.

14. Frieze, A.M., Kannan, R.and Lagarias, J.C. (1984). Linear Congruential Generators do not Produce Random Sequences. 25th Annual Symposium on Foundations of Computer Science, pp. 480-484.
15. Hastad, J. and Shamir, A. (1985). The cryptographic security of truncated linearly related variables. In Proceedings of the 17th ACM Symposium on Theory of Computing (Providence, R.I., May 6-8). ACM, New York. 356- 362.

16. Haldir, R. (2004). How to crack a Linear Congruential Generator. http://www.reteam.org/papers/e59.pdf.
17. Hugo, K. (1999). How to Predict Congruential Generators. In Proceedings on Advances in cryptology(Santa Barbara California USA). 138-153. ISBN: 0-387-97317-6.
18. Lehmer, D. H. (1951). Mathematical methods in large-scale computing units. Proceedings of the Second Symposium on Large Scale Digital Computing Machinery, Harvard University Press, Cambridge, MA, 141-146.

19. L′ Ecuyer, P. (1997). Uniform Random Number Generation: A Review. Proceedings of the 29th conference on Winter Simulation conference, Atlanta Gergia USA 127-134.
20. L′ Ecuyer, P., Blouin, F., and Couture R. (1993). A search for good multiple recursive linear random number generators. ACM Transactions on Modeling and Computer Simulation, 3, 87-98.

21.Lidl, R., and Niederreiter, H. (1994). Introduction to Finite Fields and Their Applications. RevisedEdition. Cambridge University Press, Cambridge, MA.

22.Marsaglia, G. (1968). Random Numbers fall mainly in the planes. Proceedings of the National Academy of sciencesVol. IT-15, January 1969

23.Massey, J.L. (1969). Shift-register synthesis and BCH decoding. IEEE Trans. on Inform. Theory, 61,25- 28.

24.Neal R. W. (1994). The Laws of Cryptography, online book http://www.cs.utsa.edu/wagner/lawsbookcolor/laws.pdf

25.Park, K. and Miller, K. 1988. Random Numbers Generators: Good Ones are Hard to Find. Communication of the ACMvol. 31.No.10, pp 1192-1201

26.Pommerening, K. (2003). Prediction of linear congruential generators http: //www.staff.uni–mainz.de/pommeren/

27.Raja,G and Hole, P.H. (2007). Use of the Shrinking Generator in Lightweight Cryptography for RFIDhttp://www.autoidlabs.org/uploads/media/AUTOIDLABS-WP-SWNET-024.pdf

28.Rivest, R., Shamir, A., and Adleman, L .(1978). A method for Obtaining Digital Signatures andPublic-Key Cryptosystems. Communications of the ACM, 21, 120-126.

29. Reeds, J.A (1977). Cracking Random Number Generator. Cryptologia, v.1, n. 1, 20-26

30.Shamir, A. (1983). On the Generation of Crytographically Strong. ACM Transaction on Computer Sys-tems (TOCS), pp. 38-44, Volume1 Issue1. ISSN: 0734-2071

31. Stinson, D. (2006). Cryptography: Theory and Pratice 3rd edition CRC Press, Boca Raton, Florida, ISBN:
1-58488-508-4.

32. Takashi, K., Li-Ming, W. and Niro, Y. (1996). On a nonlinear congruential pseudorandom number generator Mathematics of Computation, Volume 65, Issue 213, 227 - 233, ISSN: 0025-5718 .

33. Tausworthe, R.C. (1965). Random numbers generated by linear recurrence modulo two Mathematics ofComputation, 19: 201-209.