Nonce Generation For The Digital Signature Standard

Raj S. Katti and Rajesh G. Kavasseri
(Corresponding author: Raj S. Katti)

Department of Electrical and Computer Engineering
North Dakota State University, Fargo, ND 58105-5285, USA
(Email: {rajendra.katti, rajesh.kavasseri}@ndsu.edu)
(Received Dec. 2, 2008; revised and accepted Feb. 9, 2009)

Abstract

Digital Signature Algorithm (DSA) is an underlying algorithm to form a signature in the Digital Signature Standard (DSS). DSA uses a new random number (or nonce) each time a signature is generated for a message. In this paper, we present a Linear Congruential Generator (LCG) based approach to generate nonce for DSS. LCG has been shown to be insecure for nonce generation. If two message-signature pairs are known along with the parameters of the LCG used to generate the nonce then the private key in the signature scheme can be found, with high probability, by solving three congruences over different moduli. We use a comparison of the output of two LCGs to generate the nonces and show that our approach is secure. We also show that coupled multiple recursive generators which are similar to LCGs are also safe for nonce generation. Congruences can no longer be set up to solve for the private key. The advantage of LCG based schemes for pseudo-random number generation is their efficiency.

Keywords: Digital signature algorithm, linear congruential generator, nonce

1 Introduction

In the Digital Signature Standard (DSS) [16], it is recommended that the random number or nonce be obtained using a pseudo-random generator based on SHA-1 or DES [4, 8, 9, 10]. However these methods are computationally intensive. The vulnerability of DSS to lattice based attacks has been studied in [2, 7]. In [2], it is shown that if the nonce is generated using less computationally intensive methods like Linear Congruential Generators (LCGs) then the Digital Signature Algorithm (DSA) can be broken. The secret key can be found when the signature of two messages is known and their respective nonces are obtained as two consecutive integers generated by an LCG. This leads to three simultaneous congruences in different moduli. Such a system can be solved in polynomial time through a lattice reduction approach using Babai's nearest vector algorithm [2]. Additionally, it is shown that such an attack is applicable even if truncated LCGs are used for nonce generation. In [7], a similar approach (lattice based reduction using Babai's algorithm) was used to recover the secret (or private) key used in the DSS, provided, a sufficient number of signatures and bits of the corresponding nonces are known. Hence a weak system for nonce generation compromises the security of the overall scheme, even if the scheme is innately sound.

In this paper, we propose the use of a computationally efficient system namely the comparative or Coupled LCGs (CLCGs) for nonce generation in DSS. The result is that lattice reduction based attacks are rendered ineffective because CLCGs involve the solving of **inequalities** modulo m (m is some modulus). We show that solving such inequalities requires exponential time in the size of the modulus. Thus CLCGs are a secure, yet inexpensive method to generate nonces for the DSA. In what follows we first describe the work in [2] which shows that generating nonces using a single, or truncated LCG is insecure. We then describe our new method of generating nonces in Section 3. In Section 4 we describe why our method is secure and why lattice methods fail to break the DSA. In Section 5 we show that the complexity of obtaining the seed for the new nonce generation procedure of Section 3 is exponential. In this section we also consider a variant of our procedure for nonce generation and show that it is insecure. In Section 6 we further improve the generation of nonces by using coupled multiple recursive generators. The conclusions are collected in Section 7.

2 Preliminaries

DSS is based on the DSA. We describe the standard and the underlying DSA below.

2.1 DSA and LCG

Let $p > 2^{511}$ be a prime such that the discrete log problem in $\mathbb{Z}_p$ is intractable. Let q be a prime such that $2^{159} < q < 2^{160}$, and q divides $(p-1)$. Let $g \in \mathbb{Z}_p^*$ be of order q. Such an element can be chosen as $g = h^{(p-1)/q}$, where h is a generator of $\mathbb{Z}_p^*$. The private key x is a random integer such that $0 \leq x \leq q-1$, and $y = g^x \bmod p$. The public key is given by (p, q, y, g).
A message M is signed as follows. The signer generates a secret random number k, the nonce, such that $0 \leq k \leq q-1$. The signature (r, s) is then given by,

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ s &= (\text{SHA-1}(M) + rx)k^{-1} \bmod q. \end{aligned} \quad (1)$$

If r or s is equal to 0, a new random value of k is chosen. Note that each message has a different value for k and SHA-1 is a hash function. Verification can be done by performing the following computations.

$$\begin{aligned} u_1 &= \text{SHA-1}(M)s^{-1} \bmod q \\ u_2 &= rs^{-1} \bmod q, \\ &\text{Check if } (g^{u_1}y^{u_2} \bmod p) \bmod q = r. \end{aligned}$$

In this work we assume that the values of SHA-1(M) can be computed by the forger. Next we define LCGs and then consider the case when the nonce k is generated by an LCG.

A LCG is defined by the recurrence $(x_{i+1} = ax_i + b \bmod m)$, where a, b and m are known and x_0 is secret [14, 21]. The LCG is full period if the period of the sequence generated is m. The LCG has a fixed point (this implies that there exists i such that $x_{i+1} = x_i$) when $(1-a)^{-1} \bmod m$ exists. When this occurs the maximum period of the sequence is $m-1$, if the fixed point is not used as an initial condition. The maximum period occurs when the following conditions are satisfied.

1) b and m are relatively prime.

2) $(a-1)$ is divisible by every prime factor of m.

3) $(a-1)$ is divisible by 4 if 4 divides m.

Shamir and Hastad [6] have shown that it is possible to recover the seed x_0 if at least 1/3 of the leading bits of 3 consecutive numbers in the sequence are known. The problem of recovering the seed has also been considered in [3, 15].

Let us generate the signatures for two messages M_1 and M_2. Let the two nonces, k_1 and k_2, for each message be generated using two consecutive outputs of an LCG. Thus $k_1 = x_i$, $k_2 = x_{i+1}$ and $k_2 = ak_1 + b \bmod m$. From Equation (1) we can write,

$$\begin{aligned} s_1k_1 - r_1x &= \text{SHA-1}(M_1) \bmod q \\ s_2k_2 - r_2x &= \text{SHA-1}(M_2) \bmod q. \end{aligned}$$

The above two equations along with,

$$k_2 = ak_1 + b \bmod m,$$

form a system of simultaneous congruences with different moduli, with unknowns, k_1, k_2 and x, if messages M_1, M_2 and their signatures (r_1, s_1) and (r_2, s_2) are known. The parameters a, b and m of the LCG are also assumed to be known. The three simultaneous congruences can be solved using Babai's nearest vector algorithm and with high probability the solution yields the secret key x because the chances of getting a false solution is minimal if SHA-1(M_1) and SHA-1(M_2) are random (see Lemma 3.1 of [2]). Since the output of a hash function can be considered as random, obtaining the correct x is highly probable.

2.2 Solving the Equations

We now show how the following three equations are solved in [2].

$$\begin{aligned} s_1k_1 - r_1x &= \text{SHA-1}(M_1) \bmod q \\ s_2k_2 - r_2x &= \text{SHA-1}(M_2) \bmod q \\ -ak_1 + k_2 &= b \bmod m. \end{aligned} \quad (2)$$

Lemma 3.1 in [2] states that if $1/2 < m/q < 2$, the above system of equations have only a few solutions. We now briefly discuss the lattice based algorithm to solve the above system of equations that is given in [2]. For a survey of lattice based cryptanalysis techniques, refer to [11, 17]. Let $B = \{b_1, b_2, \cdots, b_n\}$ be a finite set of vectors in $\mathbb{R}^n$. All integer combinations of the vectors in B form a lattice denoted by $L(B)$. Finding a vector in the lattice that is close to a given vector $T \in \mathbb{R}^n$, not in the lattice is called the nearest lattice vector problem. More formally we want to find a lattice vector Z such that

$$\| Z - T \| = \min_{V \in L(B)} \| V - T \| .$$

Babai's nearest lattice vector algorithm [1], is a polynomial time approximation algorithm that finds such a vector Z, given T and B, such that

$$\| Z - T \| \leq c \times \min_{V \in L(B)} \| V - T \|,$$

where $c = 2^{n/2}$. To solve the system of Equation (2), we consider the lattice, L, generated by the columns of the following matrix.

$$B = \begin{bmatrix} -r_1 & s_1 & 0 & q & 0 & 0 \\ -r_2 & 0 & s_2 & 0 & q & 0 \\ 0 & -a & 1 & 0 & 0 & m \\ \gamma_x^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma_{k_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma_{k_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

In the above matrix $\gamma_x = min(x', m - x')$, and $\gamma_{k_1} = min(k_1', m - k_1')$ and $\gamma_{k_2} = min(k_2', m - k_2')$ where (x', k_1', k_2') are guesses for (x, k_1, k_2). Multiplying the columns of B by (x, k_1, k_2) we obtain the following lattice vector.

$$X = (M_1, M_2, b, \frac{x}{\gamma_x}, \frac{k_1}{\gamma_{k_1}}, \frac{k_2}{\gamma_{k_2}})^T. \quad (3)$$

From this lattice vector we can obtain the private key x. Our procedure consists of running Babai's nearest lattice vector algorithm on $L(B)$ and target vector

$$T = (M_1, M_2, b, \frac{x'}{\gamma_x}, \frac{k_1'}{\gamma_{k_1}}, \frac{k_2'}{\gamma_{k_2}})^T, \quad (4)$$

and obtaining a lattice vector Z such that

$$\| T - Z \| < \delta \| T - X \|,$$

where $\delta > \frac{1+c\sqrt{3}}{2}$ and $c > 2^{(3+3)/2}$ (x', k_1' and k_2' are guesses for x, k_1 and k_2 and are in the set D defined below). This follows from the fact that we have 3 equations and 3 unknowns and Lemmas 4.2 and 4.3 in [2]. In [2] Babai's algorithm was used to solve m modular equations in n variables, each with a different modulus M_i. In their formulation $\delta > \frac{1+c\sqrt{n}}{2}$ and $c > 2^{(m+n)/2}$. From Lemma 4.3 of [2] a solution (x, k_1, k_2) can be found by searching through the entire set D where $D = D_1 \times D_2 \times D_3$ and $D_i = \{\pm(1-(1-\frac{1}{\delta})^j)\frac{m_i}{2} | j = 0, 1, \cdots, \delta \log_2 \frac{m_i}{2}\}$ and $\delta > \frac{1+2^3\sqrt{3}}{2}$. $m_1 = q$, $m_2 = m_3 = m$. Therefore the number of target vectors that need to be tried to guarantee finding a solution is a polynomial in $\log_2 q$ and $\log_2 m$. This approach generalizes to truncated LCGs as well. In the next section, we introduce the proposed model (comparative LCGs) for nonce generation and describe its basic properties.

3 Comparative Linear Congruential Generators

Inspired by the concept of coupled chaotic maps [18, 19, 20], we propose a comparative LCG (CLCG) which is defined as follows:

$$\begin{aligned} x_{i+1} &= ax_i + b \bmod m \\ y_{i+1} &= cy_i + d \bmod m \\ z_{i+1} &= \begin{cases} 1 & \text{if } x_{i+1} > y_{i+1} \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Example 1. *Let $a = 5$, $b = 5$, $c = 5$, $d = 3$, and $m = 8$. Both sequences, x_i and y_i have a period of 8 and are hence full period. If the initial condition (or the seed) is $(x_0, y_0) = (2, 7)$, then the sequences are,*

$$\begin{aligned} \{x_i\} &= (7, 0, 5, 6, 3, 4, 1, 2) \\ \{y_i\} &= (6, 1, 0, 3, 2, 5, 4, 7). \end{aligned}$$

The bit sequence z_i therefore is

$$\{z_i\} = (1, 0, 1, 1, 1, 0, 0, 0).$$

We consider the problem of determining the initial condition or seed (x_0, y_0) of coupled LCGs given the bit sequence $\{z_i\}$. Note that in the computation of z_{i+1}, x_{i+1} and y_{i+1} are positive integers between 0 and $(m-1)$. This is important because computing x_{i+1} and y_{i+1}, given z_{i+1} becomes harder when x_{i+1} and y_{i+1} are restricted to being positive. We assume that a, b, c, d, m are known and the seed (x_0, y_0) is secret which leads us to the following problem.

The CLCG Problem:

Given. a, b, c, d, m and u bits of the output bit sequence, $(z_1, z_2, \cdots, z_u)$, of the coupled LCG system.

Find. The initial condition (x_0, y_0).

While we present an algorithm to solve the CLCG problem in Section 5, here, we note a few basic properties of the CLCG system that are required in the solution.

It is easy to see that the k^{th} output of an LCG $x_{i+1} = ax_i + b \bmod m$, is given as,

$$x_k = a^k x_0 + b \sum_{i=0}^{k-1} a^i \bmod m.$$

This implies that if the k^{th} output of the coupled LCGs is z_k, then the following inequality holds based on whether z_k is 1 or 0.

$$a^k x_0 + b \sum_{i=0}^{k-1} a^i \bmod m > c^k y_0 + d \sum_{i=0}^{k-1} c^i \bmod m \quad if\ z_k = 1$$

$$a^k x_0 + b \sum_{i=0}^{k-1} a^i \bmod m \leq c^k y_0 + d \sum_{i=0}^{k-1} c^i \bmod m \quad if\ z_k = 0.$$

Since u bits of the output z_k are known, we can set up u inequalities E_k, $1 \leq k \leq u$, where E_k is an inequality of the form described above.

Example 2. *For the coupled LCG system of Example 1, the inequalities E_k, $k = 1, 2, \cdots, 7$ are,*

$$\begin{aligned} 5x_0 + 5 \bmod 8 &> 5y_0 + 3 \bmod 8 \\ x_0 + 6 \bmod 8 &\leq y_0 + 2 \bmod 8 \\ 5x_0 + 3 \bmod 8 &> 5y_0 + 5 \bmod 8 \\ x_0 + 4 \bmod 8 &> y_0 + 4 \bmod 8 \\ 5x_0 + 1 \bmod 8 &> 5y_0 + 7 \bmod 8 \\ x_0 + 2 \bmod 8 &\leq y_0 + 6 \bmod 8 \\ 5x_0 + 7 \bmod 8 &\leq 5y_0 + 1 \bmod 8. \end{aligned}$$

Let S_k denotes the set of solutions (x_i, y_i) to inequality E_k. The intersection of all the S_k's for $k \in [1, u]$ gives us a small set of possible values for the seed.

Example 3. *The solution set S_1 to the first inequality is,*

$$\begin{aligned} S_1 = \{&(0,0), (0,1), (0,6), (0,3), (0,5), (1,1), (1,6), (2,1), \\ &(2,6), (2,3), (2,0), (2,5), (2,2), (2,7), (3,1), (3,6), \\ &(3,3), (3,0), (4,1), (5,1), (5,6), (5,3), (5,0), (5,5), \\ &(5,2), (6,1), (6,6), (6,3)\}. \end{aligned}$$

The intersection of the solution sets for E_1 and E_2 (the first two inequalities of Example 2) is given by,

$$\begin{aligned} S_1 \cap S_2 &= \{(0,5),(2,0),(2,1),(2,2),(2,3),(2,5),\\ &\quad (2,6),(2,7),(3,0),(3,1),(3,3),(4,1),\\ &\quad (5,1),(5,3),(5,5),(6,3)\}. \end{aligned}$$

The following lemmas, corollaries and theorems about CLCGs are stated here from [12].

Lemma 1. *If two LCGs $x_{i+1} = ax_i + b \bmod m$ and $y_{i+1} = cy_i + d \bmod m$, have full period then the inequality $ax_i + b \leq cy_i + d \bmod m$ has $\frac{m(m+1)}{2}$ solutions for (x_i, y_i).*

Corollary 1. *If two LCGs $x_{i+1} = ax_i + b \bmod m$ and $y_{i+1} = cy_i + d \bmod m$, have full period then the inequality $ax_i + b > cy_i + d \bmod m$ has $\frac{m(m-1)}{2}$ solutions for (x_i, y_i).*

Theorem 1. *Let (x_0, y_0) be a solution for inequality E_i, then the probability that it is a solution for inequality E_{i+1} is $1/2$.*

Corollary 2. *The cardinality of the intersection of the solution sets of equations $E_1, E_2, \cdots, E_u$ is $\frac{|S_1|}{2^{u-1}}$.*

Theorem 2. *The number of consecutive output bits, u, of the coupled LCGs that must be known in order to determine a unique seed is given by, $\log_2 m(m-1) \leq u \leq \log_2 m(m+1)$.*

These results imply that finding the seed to the CLCG system requires the solution of $\approx m^2 \log(m)$ congruences. We exploit this property in setting up the nonce generation scheme for DSS as described in the following section.

4 DSA and CLCG

In this section we show that if the nonce in the digital signature algorithm is generated using a CLCG, the lattice method of [2] (as explained in Section 2.2) cannot be used to find the secret key. When the CLCG is used in conjunction with the DSS, the overall system cannot be described in terms of a set of modular linear equalities, but at best, by a set of *modular linear inequalities*. The complexity of a lattice attack in this case is analyzed and shown to be prohibitively expensive, in contrast to polynomial complexity obtained with LCGs in [2].

We now propose a scheme where the nonce is obtained from r consecutive bits of a bit stream sequence that is generated by the CLCG system.

For cryptanalysis, we make the following assumptions.

A1. As in [2], the cryptanalyst knows a pair of messages M_1, M_2 and their corresponding signature pairs (r_1, s_1) =DSA(x, k_1, M_1) and (r_2, s_2) =DSA(x, k_2, M_2). Note that computing the signature using Equation (1) with private key x, nonce k_1, and message M_1 is denoted $DSA(x, k_1, M_1)$.

A2. The nonces k_1 and k_2 (r bits each) are computed from $2r$ consecutive bits governed by the CLCG system. We also assume that the starting position (index of the bit stream sequence) is known.

A3. $r \geq \log_2 m(m+1)$ (refer to Theorem 2, which guarantees a unique solution (x_0, y_0) is obtained to the CLCG system).

A4. The parameters of the CLCG system a, b, c, d and m are known and the seed (x_0, y_0) is kept secret.

Let $\{z_i\}_{i=1\cdots m-1}$ denote one period of the binary bit stream sequence generated by the CLCG system. Then the assumptions listed above yield the following set of equations:

$$\begin{aligned} s_1 k_1 - r_1 x &= M_1 \bmod q, \\ s_2 k_2 - r_2 x &= M_2 \bmod q. \end{aligned} \tag{5}$$

$$\begin{aligned} k_1 &= \sum_{i=1}^{r} z_i 2^{r-1}, \\ k_2 &= \sum_{i=r+1}^{2r} z_i 2^{r-1}. \end{aligned} \tag{6}$$

The z_i in the equation above is given by the following equations.

$$(a^k x_0 + b\sum_{i=0}^{k-1} a^i) \bmod m \models (c^k y_0 + d\sum_{i=0}^{k-1} c^i) \bmod m, \quad 1 \leq k \leq 2r, \tag{7}$$

$z_i = 1$ if $\models$ is $>$ and $z_i = 0$ if $\models$ is $\leq$.

Since the seed (x_0, y_0) is unknown, the binary sequence $\{z_i\}$ is unknown and hence, a third independent equation relating k_1 and k_2 (akin to the third equation in Equation (2)) cannot be written. Therefore, Equations (5 - 7) do not readily lend themselves to a lattice formulation. The only strategy for the opponent in this case is to solve for the seed (x_0, y_0) of the CLCG system and verify that the seed so obtained is consistent across a pair a messages (M_1, M_2). This procedure is summarized below.

1) For every one of the 2^r possibilities of k_1 do the following. Let the bits of k_1 be $\{z_i\}, i = 1, 2, \cdots, r$.

   a. Using the bits of k_1, $\{z_i\}, i = 1, 2, \cdots, r$, solve the CLCG problem thus obtaining the seed (x_0, y_0). This involves solving the following equations.

   $$(a^k x_0 + b\sum_{i=0}^{k-1} a^i) \models (c^k y_0 + d\sum_{i=0}^{k-1} c^i) \bmod m, \quad 1 \leq k \leq r.$$

   If $z_i = 1$ then $\models$ is $>$ and if $z_i = 0$ then $\models$ is $\leq$.

   b. Use this (x_0, y_0) and the CLCG system (Equation (7)) to generate $2r$ bits, $\{z_i\}, i = 1 \cdots 2r$. Bits $\{z_i\}, i = 1, 2, \cdots, r$ represent k_1 (these bits are already known) and bits $\{z_i\}, i = r+1 \cdots 2r$ represent k_2.

c. Substitute these k_1 and k_2 into Equation (5), and solve each one of these equations for x.

d. If the x's calculated in each case are the same then k_1 and k_2 are valid solutions to Equations (5) and (6). In this case stop.

2) Output the valid k_1 and k_2 that satisfy Equations (5) and (6).

Instead of checking if x computed by Equation (5), are the same, one could eliminate x from these two equations to obtain, $k_2 = \alpha_0 + \alpha_1 k_1$, where $\alpha_0 = s_2^{-1}m_2 - r_2 r_1^{-1} m_1$, $\alpha_1 = s_2^{-1} r_2 r_1^{-1} s_1$. Then one could instead check whether k_1 and k_2 of Step (b) above satisfy $k_2 = \alpha_0 + \alpha_1 k_1$. Since there are 2^r possibilities for k_1, the above procedure has to be run at most 2^r times to extract the seed (x_0, y_0) and the key x. Therefore the complexity of this procedure is equal to 2^r times the complexity of solving the CLCG Problem in Step (a) above. Therefore, the security of DSS encryption is significantly strengthened, despite using LCGs for nonce generation.

In the next section, we first give a naive method and then a lattice based method to solve the CLCG problem. We find that the lattice based methods are better than the naive method but both take exponential time. This exponential nature of the CLCG problem implies that breaking the DSA that uses CLCG for nonce generation is also exponential.

5 Solutions to the CLCG Problem

5.1 The Naive Method

In what follows we will show that finding the seed of a coupled LCG system requires an exhaustive search through the m^2 possible choices for the seed (x_0, y_0). We then consider a variant of our method to generate nonces and show how Babai's algorithm can be used to find the private key of the DSA.

Assume that we are attempting to solve the u inequalities $E_1, E_2, \cdots, E_u$, for the unique seed (x_0, y_0), where m is the modulus in the inequalities. Note that u satisfies Theorem 2. Thus we seek (x_0, y_0) that satisfies the following inequalities.

$$(a^k x_0 + b\sum_{i=0}^{k-1} a^i) \bmod m \models (c^k y_0 + d\sum_{i=0}^{k-1} c^i) \bmod m, \quad 1 \le k \le u. \tag{8}$$

Note that if $z_k = 1$ then $\models$ is $>$ and if $z_k = 0$ then $\models$ is $\le$. In the above inequalities we denote a^k by a_k, $b\sum_{i=0}^{k-1} a^i$ by b_k, c^k by c_k and $d\sum_{i=0}^{k-1} c^i$ by d_k, giving us the following inequalities.

$$(a_k x_0 + b_k) \bmod m \models (c_k y_0 + d_k) \bmod m, \quad 1 \le k \le u. \tag{9}$$

Note that $\models$ is either $\le$ or $>$ and is defined over positive integers and the modulus operation results in a positive integer less than m. One way to solve the inequalities of Equation (8) is to convert them into equalities and then to congruences as follows. These inequalities can be rewritten as an equality as follows.

$$(a_k x_0 + b_k) \bmod m = ((c_k y_0 + d_k) \bmod m + h_k) \bmod m, \quad 1 \le k \le u.$$

In the above equation $h_k < m$. The above equation can be converted into the following congruences.

$$(a_k x_0 + b_k) \equiv (c_k y_0 + d_k) + h_k \bmod m, \quad 1 \le k \le u. \tag{10}$$

These congruences now have new unknowns h_k, $k = 1, 2, \cdots, u$. Therefore we have u congruences and $u+2$ unknowns, $(x_0, y_0, h_1, \cdots, h_u)$. There is no way to solve these congruences but to guess two of the unknowns, say (x_0, y_0), and then solve for the remaining unknowns, $(h_1, h_2, \cdots, h_u)$. Assume that all the h_k are positive. After solving for these unknowns, we have to check if the $(h_1, h_2, \cdots, h_u)$, satisfy the original Inequalities (9). If they do not then it implies that the values chosen for (x_0, y_0) were incorrect. The check is performed as follows.

$$(a_k x_0 + b_k) \bmod m - (c_k y_0 + d_k) \bmod m = \begin{cases} h_k & \text{if inequality } k \text{ is } > \\ h_k - m & \text{if inequality } k \text{ is } \le \end{cases}$$

If the above condition is valid then h_k is a valid quantity that makes inequality k of Equation (9) into an equality. This is illustrated in the example below.

Example 4. *For the coupled LCG system of Example 1 the inequalities E_k, $k = 1, 2, \cdots 7$ are,*

$$\begin{aligned} 5x_0 + 5 \bmod 8 &> 5y_0 + 3 \bmod 8 \\ x_0 + 6 \bmod 8 &\le y_0 + 2 \bmod 8 \\ 5x_0 + 3 \bmod 8 &> 5y_0 + 5 \bmod 8 \\ x_0 + 4 \bmod 8 &> y_0 + 4 \bmod 8 \\ 5x_0 + 1 \bmod 8 &> 5y_0 + 7 \bmod 8 \\ x_0 + 2 \bmod 8 &\le y_0 + 6 \bmod 8 \\ 5x_0 + 7 \bmod 8 &\le 5y_0 + 1 \bmod 8. \end{aligned}$$

First we convert the above inequalities to the following congruences.

$$\begin{aligned} 5x_0 + 5 &\equiv 5y_0 + 3 + h_1 \bmod 8 \\ x_0 + 6 &\equiv y_0 + 2 + h_2 \bmod 8 \\ 5x_0 + 3 &\equiv 5y_0 + 5 + h_3 \bmod 8 \\ x_0 + 4 &\equiv y_0 + 4 + h_4 \bmod 8 \\ 5x_0 + 1 &\equiv 5y_0 + 7 + h_5 \bmod 8 \\ x_0 + 2 &\equiv y_0 + 6 + h_6 \bmod 8 \\ 5x_0 + 7 &\equiv 5y_0 + 1 + h_7 \bmod 8. \end{aligned}$$

Let us check if $(x_0, y_0) = (0, 0)$ *is a solution to the above inequalities. When* $(x_0, y_0) = (0, 0)$, $(h_1, h_2, \cdots, h_7) = (2, 4, 6, 0, 2, 4, 6)$. $h_1 = 2$ *is a valid solution because*

$$(5x_0 + 5) \bmod 8 - (5y_0 + 3) \bmod 8 = 5 - 3 = 2 = h_1.$$

Similarly $h_2 = 4$ *is not a valid solution because*

$$(x_0+6)-(y_0+2) \bmod 8 = 6-2 = 4 \neq h_2-m = 4-8 = -4.$$

Therefore $(x_0, y_0) = (0, 0)$ *is not a valid solution for above inequalities.*

It should be noted that the comparison operator is defined over positive integers, which are generated by the modulus operation in each LCG. This makes the search for a seed exponential. To appreciate this, consider the first inequality in Example 1: $(5x_0 + 5) \bmod 8 > (5y_0 + 3) \bmod 8$. We wish to point out that solutions to this inequality cannot be obtained by merely manipulating it to: $5x - 5y + 2 > 0 \bmod 8$. For example, $(x, y) = (1, 0)$ satisfies this inequality while violating $(2 < 3)$ the original inequality, where the left and right hand sides evaluate to 2 and 3 respectively. However, $(x, y) = (1, 0)$ can be made to obey the original inequality if the right hand side 3 is set to $-5 \bmod 8$.

Thus a search through the entire solution space is needed to find the unique (x_0, y_0) that satisfies Inequalities (9). On the average we will have to try out $m^2/2$ values for (x_0, y_0) before arriving at the correct solution, therefore the complexity of the above procedure is $m^2/2 \times C_c$, where C_c denotes the complexity of solving u congruences in u unknowns.

The congruences of Equation (10) all have the same modulus and can therefore be solved using Gaussian elimination instead of lattice methods, which are usually invoked only when the moduli are different. However, the lattice method can still be used with the CLCG system if the nonces are chosen to be the differences of the outputs of the individual LCGs themselves. In this case, three additional message-signature pairs will suffice to mount a lattice attack with polynomial complexity, as explained below.

Suppose the nonces chosen (h_k) are given by:

$$((a_k x_0 + b_k) - (c_k y_0 + d_k)) \bmod m = h_k,\ 1 \leq k \leq u.$$

Even though manipulation of the inequalities is forbidden, we convert the congruences of Equation (10) to the following congruences. Such a manipulation may yield solutions that are incorrect, implying that we must verify if every solution obtained using these equations satisfies the original inequalities of Equation (9).

$$\begin{aligned} a_1x_0 - c_1y_0 - h_1 &\equiv d_1 - b_1 = w_1 \bmod m \\ &\vdots \\ a_ux_0 - c_uy_0 - h_u &\equiv d_u - b_u = w_u \bmod m. \end{aligned} \quad (11)$$

In the above equalities a_i, b_i, c_i, d_i, $(w_1, w_2, \cdots, w_u)$ and m are known and $x_0, y_0, h_1, \cdots, h_u$ are unknowns. If three message-signature pairs are known then three more equations can be set up as follows.

$$\begin{aligned} s_1h_1 - r_1x &= M_1 \bmod q \\ s_2h_2 - r_2x &= M_2 \bmod q \\ s_3h_3 - r_3x &= M_3 \bmod q. \end{aligned} \quad (12)$$

Note that h_1, h_2, h_3 are the nonces. Therefore, Equations (11) and (12) form a system of $(u + 3)$ congruences with $(u+3)$ unknowns $x_0, y_0, h_1, \cdots, h_u, x$. From Lemma 3.1 in [2] and since $u > \log_2 m(m + 1)$, it follows that these equations have a small number of solutions for the unknowns. These equations have different moduli and therefore can be solved using Babai's nearest vector algorithm. From the preliminaries section we know that such a method is polynomial in $\log_2 q$ and $\log_2 m$. Therefore even if Equation (11) have m^2 solutions, lattice methods can be used once again to find the solutions to both Equations (11) and (12). Thus using such h_k as nonces is insecure.

Our original method generates bits of a nonce by generating a 1 if $(a_k x_0 + b_k) \bmod m > (c_k y_0 + d_k) \bmod m$ and a 0 otherwise. This makes it very difficult to use lattice methods for obtaining the secret key in the digital signature algorithm.

5.2 The Lattice Method

The naive method of the previous sub-section for solving the CLCG problem results in a time complexity of $\mathcal{O}(m^2)$ with memory requirement of $\mathcal{O}(u) \approx \mathcal{O}(\log m)$ (for the $2 \log m$ equations that need to be solved). We now present another method of solving the CLCG problem that has time complexity of $\mathcal{O}(m \log m)$ but with a memory requirement that is greater than a polynomial in $\log m$. This method works by first converting the inequalities in two variables into m sets of inequalities in one variable, one set for each value of y_0. Therefore for each $y_0 = 0, 1, \cdots, (m - 1)$ perform the following three steps.

Step 1.

In this step we obtain u equalities from the m inequalities similar to Equation (8). These inequalities are stated once again below. Let the output of the CLCG system be z_k for $k = 1, 2, \cdots, (m - 1)$. The inequalities corresponding to bits z_k are as follows.

$$(a_k x_0 + b_k) \bmod m \models (c_k y_0 + d_k) \bmod m, \quad 1 \leq k \leq (m - 1). \quad (13)$$

In the above equation, $\models$ is $\leq$, if the corresponding output bit z_k is 0, and is $>$ otherwise. For each inequality substitute the current value of $y_0 = i$, $0 \leq i \leq (m - 1)$ and compute the right hand side to obtain an integer less than m. Then select a set of $u \geq p(\log m)$ from the above $(m - 1)$ inequalities such that the right hand side computed is an integer

close to 0 or m. Here $p(\log m)$ is some polynomial in $\log m$. This results in inequalities similar to one of the following two inequalities.

$$\begin{aligned}(a_k x_0 + b_k) \bmod m &\leq Z \\ (a_k x_0 + b_k) \bmod m &\geq G. \end{aligned} \quad (14)$$

In the above inequalities Z is an integer close to 0 and G is an integer close to m. By "close to" we mean that $Z \leq p_0(\log m)$, and $(m - G) \leq p_1(\log m)$, where $p_0(\log m)$ and $p_1(\log m)$ are polynomials in $\log m$. We can approximate Z and G as 0 *mod* m resulting in the following set of equalities that can be solved using lattice methods. Note that lattice methods are useful here because Z and G are approximated as 0 to obtain the approximate equations below.

$$\begin{aligned}(a_k x_0 + b_k) \bmod m = 0, \\ 0 \leq k \leq (u-1),\ u \geq 2\log m.\end{aligned}$$

One of the solutions to the above equations could be a solution to the inequalities of Equation (14) above. For this step to have constant time complexity we need to generate all the inequalities of Equation (13) in one time unit (thus requiring at most $\mathcal{O}(m)$ memory locations), and then substitute the value of y_0 in the right hand side of these inequalities in order to obtain Equation (14). If we did this one inequality at a time then this process would take $\mathcal{O}(m)$ time steps thus requiring a constant number of memory locations. To confirm this we prove the following theorem that states that it is *not* possible to obtain Equation (14) by scanning only a polynomial number of inequalities in Equation (13).

Theorem 3. *If $u = p(\log m)$, a polynomial in $\log m$, then the number of inequalities of Equation (13) that have to be scanned before obtaining u inequalities like in Equation (14) is more than $g(\log m)$, where $g(\cdot)$ is any polynomial.*

Proof. Let us assume that we have to compute the right hand side of $q(\log m) > u$ inequalities of Equation (13) in order to obtain u equations like the ones in Equation (14) (that is the right hand side is close to either 0 or m, where $q(\log m)$ is a polynomial. Since the right hand side is an LCG the probability of obtaining any integer is $1/m$. The probability of an LCG output being either Z (close to zero) or G (close to m) is $\frac{p_0(\log m)+p_1(\log m)}{m}$ ($p_0(\cdot)$ and $p_1(\cdot)$ are polynomials such that $Z \leq p_0(\log m)$ and $(m-G) \leq p_1(\log m)$). If $\frac{(p_0(\log m)+p_1(\log m))q(\log m)}{m} \geq 1$ then with certainty we can say that at least u out of $q(\log m)$ inequalities' right hand side evaluates to an integer close to 0 or m. This implies that $m \leq g(\log m)$, where $g(\cdot) = (p_0(\cdot) + p_1(\cdot))q(\cdot)$. This contradicts the fact that m cannot be a polynomial in $logm$. Therefore we cannot obtain the required u inequalities by scanning a polynomial number of inequalities in Equation (13). □

Step 2.

In this step we solve the u equalities of Step 1 using lattice methods. Consider the lattice, L, generated by the columns of the following matrix.

$$B = \begin{bmatrix} a_1 & m & 0 & \cdots & 0 \\ a_2 & 0 & m & \cdots & 0 \\ \vdots & \vdots & & \vdots & \\ a_u & 0 & 0 & \cdots & 0 \\ \gamma_{x_0}^{-1} & 0 & 0 & \cdots & 0 \end{bmatrix}$$

In the above matrix $\gamma_{x_0} = min(x_0', m - x_0')$, where x_0' is a guess for x_0. Multiplying the first column of B by x_0 we obtain the following lattice vector.

$$X = (-b_1, -b_2, \cdots, -b_u, \frac{x_0}{\gamma_{x_0}})^T.$$

From this lattice vector we can obtain x_0. Our procedure consists of running Babai's nearest lattice vector algorithm on $L(B)$ and target vector

$$T = (-b_1, -b_2, \cdots, -b_u, \frac{x_0'}{\gamma_{x_0'}})^T,$$

and obtaining a lattice vector Z such that

$$\| T - Z \| < \delta \| T - X \|,$$

where $\delta > \frac{1+c\sqrt{1}}{2}$ and $c > 2^{(u+1)/2}$ (x_0' is a guess for x_0). This follows from the fact that we have u equations and 1 unknown and Lemmas 4.2 and 4.3 in [2]. In [2] Babai's algorithm was used to solve m modular equations in n variables, each with a different modulus M_i. In their formulation $\delta > \frac{1+c\sqrt{n}}{2}$ and $c > 2^{(m+n)/2}$.

The time complexity of obtaining x_0 is therefore $\mathcal{O}(\delta \log m)$ and is $\mathcal{O}(\log m)$ if u is small. Sometimes Equation (14) can be solved in constant time. To illustrate this we consider the situation when Z in this equation is 0. Then the top inequality of Equation (14) becomes, $(a_k x_0 + b_k) \bmod m \leq 0$ implying, $(a_k x_0 + b_k) \bmod m = 0$, and $x_0 = -b_k a_k^{-1} \bmod m$. In the rest of this work we therefore assume that solving for x_0 takes constant time.

Step 3.

In this step we check if the solutions for (x_0, y_0) from Step 2 are correct. x_0 was obtained in Step 2 and y_0 took on a value before starting Step 1. This (x_0, y_0) is the correct solution if it satisfies at least $2\log m$ inequalities from Equation (13) (see Theorem 3.5). If the solution is correct we stop the algorithm. If the solution is incorrect, we go to the next y_0 and then go back to Step 1. The time complexity of this step is $\mathcal{O}(\log m)$.

We now describe the time and space complexity of the above algorithm keeping in mind that Steps 1, 2 and 3 are executed m times, once for each value of y_0. Thus the total time complexity is either $\mathcal{O}(m \times \log m)$ or $\mathcal{O}(m \times (m + \log m))$ depending on whether the number of memory locations used is either at most $\mathcal{O}(m)$ or a constant respectively.

5.3 Difficulty of the CLCG problem

We now take a brief look at why the CLCG problem is difficult. The difficulty of solving the CLCG problem is linked with the comparison operator. Solving the CLCG problem requires solving inequalities of the form specified by Equation (13). One of the best ways of solving these inequalities is to somehow convert them to equalities or congruences (such methods have been considered earlier in this section) and then use lattice-like methods to obtain the solution. One of the main difficulties in dealing with inequalities is the fact that there is no ordering over integers modulo m. This is because $x \bmod m$ can be both less than and greater than another integer y. The reason for this is the fact that y and $y - m$ are congruent. For example $3 \bmod 11$ is less than 10 and greater than $-1 = 10 \bmod 11$.

Another difficulty with an inequality of the form $ax + b \bmod m > cy + d \bmod m$ is the fact that it cannot be manipulated. This implies that this inequality cannot be converted to the following, $ax+b-cy-d \bmod m > 0 \bmod m$. Such a conversion would lead to incorrect solutions for (x, y) as we have noted earlier in this section.

Lastly we note that lattice methods for solving modular equalities lead to exponential complexity in the input size ($\log m$ is the input size).

In the following section, we demonstrate how the coupled LCG system can be extended to enhance the security of the DSS algorithm further.

6 Multiple Recursive Generators

Recall that a multiple recursive generator (MRG) (see [5, 14, 22]) is defined by:

$$x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k} \bmod m,$$

where $a_i \in \mathbb{Z}_m$. Such generators have a period of $m^k - 1$ if and only if m is prime and the polynomial $P(z) = z^k - a_1 z^{k-1} - \cdots - a_k$ is primitive [14]. We can form a coupled multiple recursive generator (CMRG) to generate a bit sequence as follows.

$$\begin{aligned} x_n &= a_1 x_{n-1} + \cdots + a_k x_{n-k} \bmod m \\ y_n &= b_1 y_{n-1} + \cdots + b_k y_{n-k} \bmod m \\ z_n &= \begin{cases} 1 & \text{if } x_n > y_n \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

In the above setting a_i and b_i are chosen such that the multiple recursive generators x_i and y_i have period $m^k - 1$. Here the seed for the coupled system is $(\mathbf{x0}, \mathbf{y0})$ where $\mathbf{x0} = (x0_1, x0_2, \cdots x0_k)$ and $\mathbf{y0} = (y0_1, y0_2, \cdots y0_k)$.

Let every r consecutive bits of z_n be chosen as a nonce for DSA. Again no lattice formulation is possible in order to obtain the private key in the DSA. The only way to break the DSA then is to first obtain the seed of the coupled MRG, with no known z_n. In order to obtain a unique seed at least $log_2 m^{2k}$ bits of z_n must be known (this is similar to Theorem 3.5 for LCGs). Since $log_2 m^{2k}$ can be very large compared to the size of a nonce, we will have to guess several nonces to obtain enough bits of z_n and then solve for the seed. Thus the procedure of Section 4 can still be used but Step 1 will have to be modified to:

1) For every one of the 2^s possibilities of several nonces, such that $s > log_2 m^{2k}$ do the following. Let the bits of these nonces be $\{z_i\}, i = 1, 2, \cdots, s$.

Lattice algorithms can once again be used if the nonces are the h_k defined by the equations below.

$$\begin{aligned} &((a_{1_i} x0_1 + a_{2_i} x0_2 + \cdots + a_{k_i} x0_k) \bmod m \\ &\quad -(b_{1_i} y0_1 + b_{2_i} y0_2 + \cdots + b_{k_i} y0_k) \bmod m) \bmod m \\ &\quad = h_i, \ 1 \le i \le s. \end{aligned}$$

In the above equation $s > log_2 m^{2k}$. If $2k + 1$ message-signature pairs are known then the following new congruences can be formed.

$$\begin{aligned} &a_{1_i} x0_1 + a_{2_i} x0_2 + \cdots + a_{k_i} x0_k \\ &\quad -(b_{1_i} y0_1 + b_{2_i} y0_2 + \cdots + b_{k_i} y0_k) - h_i \\ &\quad \equiv 0 \bmod m, \quad 1 \le i \le s. \end{aligned} \tag{15}$$

$$\begin{aligned} s_1 h_1 - r_1 x &= M_1 \bmod q \\ s_2 h_2 - r_2 x &= M_2 \bmod q \\ &\vdots \\ s_{2k+1} h_{2k+1} - r_{2k+1} x &= M_{2k+1} \bmod q. \end{aligned} \tag{16}$$

Equations (15) and (16) form a system of $2k+s+1$ congruences in different moduli with $2k + s + 1$ unknowns, $(\mathbf{x0}, \mathbf{y0}), (h_1, h_2, \cdots, h_s), x$. Once again these equations have a small number of solutions that can be found with Babai's nearest vector algorithm in polynomial time. Even though solving Equation (15) for the seed requires an exponential amount of time, using h_i as the nonces is insecure.

If we use $\mathcal{O}(m^k)$ memory locations then using the method of Section 5.2, the time complexity of finding the seed for CMRGs can be reduced to $\mathcal{O}(m^k (\delta \log m)^k)$, where $\delta = \frac{1+2^{(k+u)/2}\sqrt{k}}{2}$. Here u is small (a polynomial in $\log m$) and is the number of equations in $\mathbf{x0}$ that need to be solved for a given value of $\mathbf{y0}$. Also, Step (1) in the procedure of Section 4 requires a larger search space than the CLCG case. Thus coupled multiple recursive generators are better than coupled LCGs for nonce generation, but come at the expense of extra computation.

7 Conclusion

We have shown that coupled LCGs are good candidates for nonce generation in the digital signature standard in terms of both security and computational efficiency. The weaknesses of a single LCG are removed by the coupling. This is primarily because solving inequalities modulo m is an exponential operation in the size of m. We show that the only way to break the digital signature algorithm when coupled LCGs are used is to first solve the CLCG problem (that is find the seed for the CLCG) and then find the private key if two message-signature pairs are known. We also show that the lattice based algorithm to break the coupled LCGs has a complexity of $\mathcal{O}(n2^n)$, where $m = 2^n$. However we **cannot** achieve this with an amount of memory that is polynomial in n. Therefore the task of breaking coupled LCGs is computationally infeasible for large m. Using coupled multiple recursive generators makes nonce generation more secure because the seed space is m^{2k} as opposed to m^2 in the CLCG case. Finally we have shown that lattice based methods are rendered useless if the random numbers needed in a digital signature scheme are generated using our new coupled LCG scheme. The methods of this paper can easily be extended to non-LCGs. The methods of this paper can also be used to generate pseudo-random bit sequences that can be used for other applications. We end with a brief description of the efficiency of our method. The proposed LCG consists of two independent LCGs and a comparator. To produce one bit of the nonce, this requires two modular multiplications, and two modular additions and a comparison operation. However, choosing a modulus that is a power of 2, reduces this to only 2 shifts, 2 additions, and a comparison operation [13]. The original LCG has only one LCG and is therefore more efficient but not secure. Other nonce generation procedures use either SHA-1 or DES/AES. Both these methods consist of performing several rounds of complicated operations and are therefore less efficient than the proposed method.

References

[1] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1-13, 1986.

[2] M. Bellare, S. Goldwasser, and D. Macciancio, "Pseudo-random number generation within cryptographic algorithms: The DSS case," *Proceedings of Advances in Cryptography (Crypto'97)*, pp. 277-291, 1997.

[3] J. Boyar, "Inferring sequences produced by pseudo-random number generators," *Journal of the ACM*, vol. 36, no. 1, pp. 129-141, 1989.

[4] D. S. A. Elminaam, H. M. A. Kader, and M. M. Hadhoud, "Evaluating the performance of symmetric encryption algorithms," *International Journal of Network Security*, Vol. 10, No. 3, 2010, pp. 213-219

[5] A. Grube, *Mehrfach Rekursiv Erzeugte Zufallszahlen*, Ph.D. Thesis, University of Karlsruhe, 1973.

[6] J. Hastad and A. Shamir, "The cryptographic security of truncated linearly related variables," *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pp. 356-362, Rhode Island, 1985.

[7] N. A. Howgrave-Graham and N. P. Smart, *Lattice Attacks on Digital Signature Schemes*, Technical report, HP Labs, HPL-1999-90, Aug. 1999.

[8] H. M. H. Huseim, B. I. Bayoumi, F. S. Holail, B. E. M. Hasan, and M. Z. A. El-Mageed, "A genetic algorithm for cryptanalysis of DES-8," *International Journal of Network Security*, vol. 5, no. 2, pp. 213-219, 2007.

[9] H. M. H. Huseim, B. I. Bayoumi, F. S. Holail, B. E. M. Hasan, and M. Z. A. El-Mageed, "A genetic algorithm for cryptanalysis with application to DES-like systems," *International Journal of Network Security*, vol. 8, no. 2, pp. 177-186, 2009.

[10] M. S. Hwang and C. C. Lee, "Research issues and challenges for multiple digital signatures," *International Journal of Network Security*, vol. 1, no. 1, pp. 1-7, 2005.

[11] A. Joux and J. Stern, "Lattice reduction: A toolbox for the cryptanalyst," *Journal of Cryptology*, vol. 11, no. 3, pp.161-185, 1998.

[12] R. S. Katti and R. G. Kavasseri, "Secure pseudo-random bit generation using coupled linear congruential generators," *IEEE International Symposium on Circuits and Systems*, pp. 2929-2932, 2008.

[13] R. S. Katti and S. K. Srinivasan, "Efficient hardware implementation of a new pseudo-random bit sequence generator," *The IEEE International Symposium on Circuits and Systems*, pp. 1393-1396, Taiwan, May 2009.

[14] D. E. Knuth, *Seminumerical Algorithms, The Art of Computer Programming (Vol. 2)*, Addison-Wesley, Reading, Mass., 1969.

[15] D. E. Knuth, "Deciphering a linear congruential encryption," *IEEE Transactions on Information Theory*, vol. 83, no. 11, pp. 49-52, 1985.

[16] National Institute of Standards and Technology (NIST), *FIPS Publication 186: Digital Signature Standard*, May 19, 1994.

[17] P. Q. Nguyen and J. Stern, "Lattice reduction in cryptology: An update," *Proceedings of 4'th International Symposium on Algorithms in Number Theory (ANTS-IV)*, Lecture Notes in Computer Science, Springer-Verlag, pp. 85-112, 2000.

[18] N. K. A Pareek, V. Patidar, and K. K. Sud, "Random bit generator using chaotic maps," *International Journal of Network Security*, vol. 10, no. 1, pp. 32-38, 2010.

[19] L. Shunjun, M. Xuanqin, and C. Yuanlong, "Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography," *Proceedings of the Second International Conference on Cryptology in India*, pp. 316-329, 2001.

[20] S. Li, Q. Li, W. Li, X. Mou, and Y. Cai, "Statistical properties of digital piecewise linear chaotic maps and their roles in cryptography and pseudo-random coding," *Proceedings of the* 8^{th} *IMA International Conference*, pp. 205-221, Dec. 2001.

[21] D. Stinson, *Cryptography: Theory and Practice*, Chapman & Hall, 3^{rd} edition, 2006.

[22] R. C. Tausworthe, "Random numbers generated by linear recurrence modulo two," *Mathematics of Computation*, vol. 19, no. 90, pp. 201-209, 1965.

Raj S. Katti received the B. Tech. degree from the Indian Institute of Technology (Bombay), India in 1983. He received the M.S. in mechanical engineering from the University of Idaho in 1985, the M.S. and Ph.D. in Electrical Engineering from Washington State University in 1987 and 1991 respectively. Dr. Katti has received funding in the area of security/cryptography from the National Science Foundation and Intel Corporation. His interests are in cryptographic hardware, fault tolerant computing, computer architecture, and asynchronous circuit design. He has published over 50 journal and conference papers on these topics. He was a senior design engineer at the Intel Corporation in 2000 and 2001. He has also taught at the Wichita State University in Kansas. He has collaborated with the IBM Almaden Research Center for the development of unidirectional error correcting codes. He is currently a Full Professor in the Department of Electrical and Computer Engineering at North Dakota State University.

Rajesh G. Kavasseri received his B.E (1995) in Electrical Engineering from Visvesvaraya Regional College of Engineering, Nagpur, India, M.Sc.(1998) in Electrical Engineering from the Indian Institute of Science, Bangalore, India and Ph.D.(2002) in Electrical Engineering from Washington State University, Pullman, WA. He has been with the Department of Electrical and Computer Engineering at North Dakota State University (Fargo, ND) since 2002, where he is currently an associate professor. His research interests include computational methods in power systems analysis/dynamics and nonlinear dynamical systems. He is a senior member of the IEEE and serves as the Chair for the IEEE Red River Valley Section.