WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2011; **11**:254–266
Published online 01 July 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.988

SPECIAL ISSUE PAPER

A lightweight block cipher based on a multiple recursive generator for wireless sensor networks and RFID

Alina Olteanu[1], Yang Xiao[1]*, Fei Hu[2], Bo Sun[3] and Hongmei Deng[4]

[1] Department of Computer Science, The University of Alabama, Tuscaloosa, AL 35487, U.S.A.
[2] Department of Computer Science, The University of Alabama, AL 35487, U.S.A.
[3] Department of Computer Science, Lamar University, Beaumont, TX 77710, U.S.A.
[4] Intelligent Automation, Inc. (IAI), MD, U.S.A.

ABSTRACT

In this paper, we use a multiple recursive generator (MRG) to generate sequences of numbers with very long periods, i.e., pseudo-random sequences. The MRG effectively constructs a block cipher which satisfies important quality requirements such as security, long period, randomness, and efficiency. We compare our approach with another lightweight block cipher based on a linear congruential generator (LCG) and analyze the efficiency in terms of the number of basic operations that are being performed. We also study the effects of using special classes of MRG which hold certain portability and efficiency properties, and analyze their advantages in this context. The proposed cipher is a lightweight cipher, which is very useful for resource limited resources such as sensor nodes in sensor networks, radio frequency identification (RFID) tags, etc.

KEYWORDS
lightweight cipher; sensor network; RFID

***Correspondence**
Yang Xiao, Department of Computer Science, The University of Alabama, AL 35487, U.S.A.
E-mail: yangxiao@ieee.org

1. INTRODUCTION

Random sequences are commonly used in ciphers and key management. For some devices, such as sensor nodes in wireless sensor networks (WSNs) and radio frequency identification (RFID) tags, with limited resources in terms of computational power, bandwidth, and size, using a random generator would only imply the transfer and storage of a short seed. However, since any algorithm has an implicit/explicit finite state machine, the produced sequence is not truly random; due to the finite number of states the generated sequence becomes periodic. The period length of such a sequence (i.e., the maximum length of the sequence before it starts to repeat) is thus an important factor in achieving randomness. A shorter than expected period might lead to the sequence failing statistical pattern detection tests. Many security schemes have been proposed [17–52].

In this paper, our goal is a multiple recursive generator (MRG) as the basis of a lightweight block cipher with the purpose of satisfying important quality requirements [1] such as the security, long period, randomness, and efficiency properties. We design a new lightweight cipher by applying the MRG described in Reference [2] to an encryption scheme based on a popular, simpler pseudo-number generator, called linear congruential generator (LCG) [3]. We analyze the overhead introduced by such a change and assess the tradeoff between the improved security of the scheme and the possible loss in performance, in terms of the number of basic operations that need to be executed when using such a pseudo-number generator. We also consider special classes of MRGs which in addition to the large period property are highly efficient, portable, and exhibit good statistical properties [4–6]. We find that ciphers based on such variations of MRGs perform significantly better than the LCG-based cipher while at the same time being secure enough for WSNs.

Such lightweight block ciphers can be applied to WSNs, RFIDs, and their integration, as well as applications such as wireless telemedicine [7–10].

The rest of the paper is organized as follows. In Section 2, we present the background and significant related

work. In Section 3, we introduce our block cipher based on the MRG generator. We then give a detailed analysis of the number of operations necessary for both generating a pseudo-random number, and for the block cipher, in Section 4. In Section 5, we compare our results with the performance of the LCG-based cipher. In Section 6, we provide performance analysis. In Section 7, we introduce some special classes of MRGs and analyze their efficiency. Section 8 presents approaches of speeding MRGS. Section 9 presents applications in WSNs and RFID sensor networks while Section 10 concludes the paper.

Note that the LCG-based cipher in Reference [11] does not use LCG as the cipher, but is a cipher utilizing LCG. Since the LCG-based cipher in Reference [11] has been compared with other ciphers, in this paper, we only compare our cipher with the LCG-based cipher in Reference [11]. Since in Reference [11], we discussed some issues about the key managements, we will not consider this aspect in this paper. Furthermore, the proposed block cipher may be more suitable for active RFID instead of passive RFID. The proposed MRG-based cipher may not only have usefulness as a cipher, but also it could be used in parts of other ciphers as random number generators instead of a standalone cipher.

2. BACKGROUND AND RELATED WORK

Deng [2] presents a series of efficient and portable MRGs having very large period lengths of up to $10^{14903.1}$.

An MRG with k initial seeds, $X_0, \ldots, X_{k-1}$ is given by:

$$X_i = (\alpha_1 X_{i-1} + \ldots + \alpha_k X_{i-k}) \mod p, \quad i \geq k \qquad (1)$$

The period length is an essential measure of the randomness of the generated numbers.

Deng [2] shows that if the characteristic polynomial of the MRG in Equation (1) is a primitive polynomial (please see Reference [3] for the definition of a primitive polynomial), then the maximum period is reached and it is equal to $p^k - 1$.

Also from Reference [2], the characteristic polynomial of an MRG is given by:

$$f(x) = x^k - \alpha_1 x^{k-1} - \ldots - \alpha_k \qquad (2)$$

Primitive polynomials define a recurrence relation that can be used to generate a new pseudo-random number from the k preceding ones. For example, given $p = 2$ and the primitive polynomial$x^{10} + x^3 + 1$, over GF (2), we start with a pre-specified 10-bit seed that can be randomly chosen. From the form of the primitive polynomial, we also know the values of the first 10 coefficients $\alpha_i, i = 0, \ldots, 9$, the rest of the coefficients being 0. The seed is then xor-ed together with the coefficient vector bit by bit to obtain the next state of the generator. This process is repeated until the initial state is reached again, and can be used to generate a maximum of $p^{10} - 1 = 2^{10} - 1 = 1023$ pseudo-random bits (please see Reference [3]) corresponding to $p^{10} - 1$ different pseudo-random numbers.

In general, for a primitive polynomial of degree k, this process will generate at most $p^k - 1$ pseudo-random numbers before repeating the same sequence. $p^k - 1$ is called the period of the pseudo-random number generator. Much research has focused on finding primitive characteristic polynomials because they lead to maximum periods.

Knuth [12] gives a set of three necessary and sufficient conditions under which $f(x)$ given by Equation (2) is a primitive polynomial:

(a) $(-1)^{k-1}\alpha_k$ must be a primitive root mod p;
(b) $x^R = (-1)^{k-1}\alpha_k \mod \gcd(f(x), p)$, where $R(k, p) = \left(p^k - 1\right)/(p - 1)$;
(c) for each prime factor q of R, the degree of $x^{R/q} \mod \gcd(f(x), p)$ is positive.

The first condition can be easily checked using the definition of a primitive root mod p. An integer A is a primitive root mod p if there exists the smallest positive exponent e such that $(A^e = 1 \mod p) = p - 1$. Moreover, based on Reference [2], the number of primitive roots mod p is equal to the number of integers between 1 and $p - 1$ that are relatively prime to $p - 1$.

The second and third conditions pose some difficulty. The third condition requires the decomposition of $p^k - 1$ into prime factors, where p is a prime number. A common approach is to choose$p = 2^{31} - 1$, because it is the largest prime number that can be stored as a signed integer in a 32-bit computer word [38]. Given $p = 2^{31} - 1$, Deng [38] has found three values for $k, k = 47, k = 643$, and$k = 1597$, for which R can be easily factored, as being the product of one small prime number and a large one. Given these values for k, the next step is trying to find primitive polynomials having the degree equal to one of these three values. Using the k values above, $p^k - 1$ can be decomposed into factors and primitive polynomials of degree k can then be found.

For condition b) above, Deng [2] also presents an alternate, more efficient method of checking the primality of the characteristic polynomial associated with the MRG. This new condition provides an early exit strategy if $f(x)$ given by Equation (2) is not a primitive polynomial.

Based on Reference [2], using conditions (a) and (c) and the updated condition (b) one can find many primitive polynomials of degree 1597, and therefore obtain a very high period of approximately $10^{14903.1}$.

To achieve increased efficiency and portability, a series of special types of MRGs which also exhibit the large period properties have been studied [2,4–6][12,13]. Among them, the DX-k-s generators [2,5] with $k = 1537$and $s = 4$ achieve the same period length as the general MRG from Equation (1) ($k = 1537$) but are far more efficient having only four non-zero terms. In addition, this class of generators possesses good statistical properties, such as a suitable lattice structure [12].

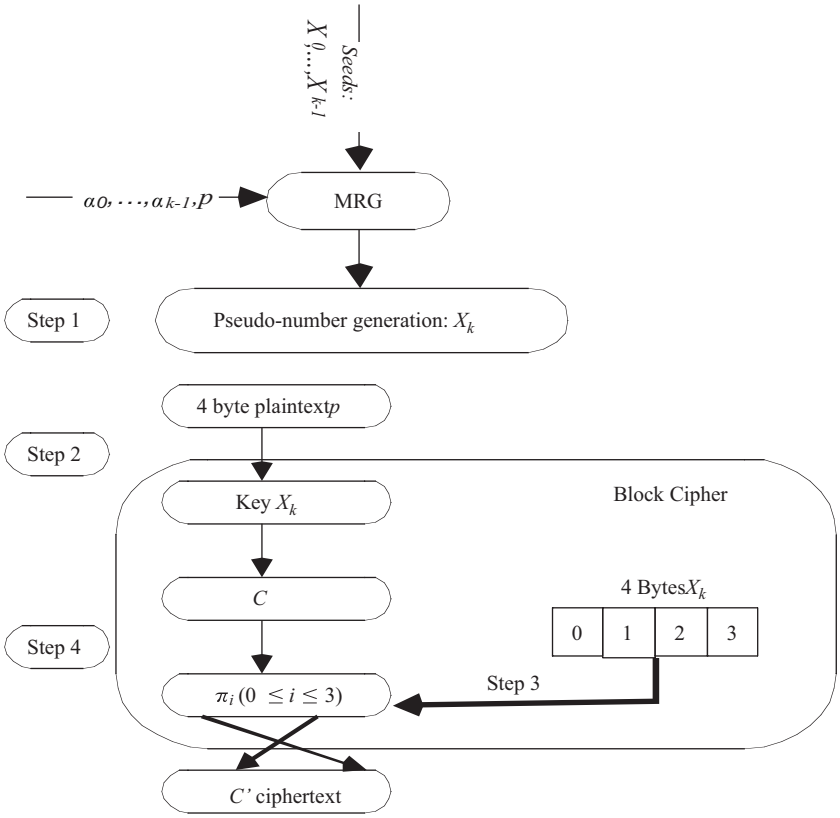


Figure 1. Message encryption of a 4 byte packet.

Paper [11] applies another type of generator, the LCG to generate pseudo-random keys used in the design of a lightweight block cipher suitable for sensor networks. The LCG is a very popular, simpler pseudo-number generator, given by:

$$x_{n+1} = ax_n + b \mod m \tag{3}$$

Although the LCG has the advantage of being very simple, it does not exhibit a large period. The period of the LCG is at most equal to the size of the modulo, m, and in most cases less than that. This means that after at most m numbers, the generated numbers will start repeating themselves. Obviously, a large modulo size will produce periods long enough for any practical application. However, in the context of limited processing power and storage which characterize WSNs and RFIDs, smaller sizes for the modulo and generator parameters can significantly increase the efficiency of encryption schemes. This is the purpose of the present paper. We show that the same or higher security level can be achieved by using classes of MRGs which have critically smaller parameter sizes but achieve significantly larger period sizes.

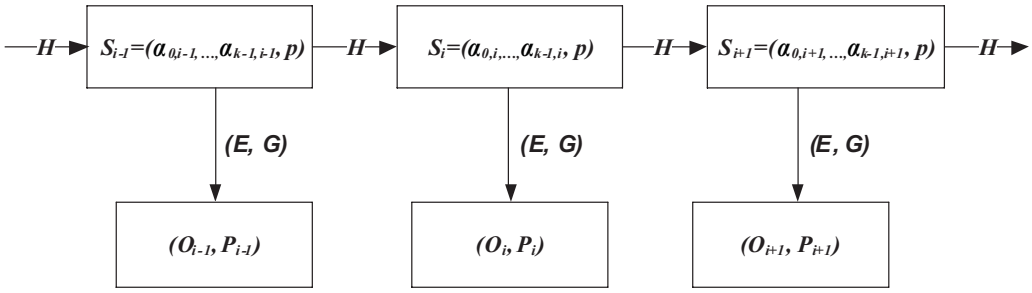
3. THE BLOCK CIPHER

3.1. The cipher

As it is suggested in Reference [14], if an adversary has access to five or more consecutive numbers generated using Equation (3), then by the use of Plumstead's Algorithm [15] the parameters of the LCG can be found, and thus the whole sequence can be discovered. The security of the cipher in Reference [14] is achieved by adding random noise and random permutations to the original data messages. Therefore, it is sufficient to just keep the seeds $X_0, \ldots, X_{k-1}$ secret while $\alpha_0, \ldots, \alpha_{k-1}$ and p can be public, as shown in Figure 1.

Likewise, if we substitute the LCG in Equation (3) with our MRG from Equation (1), it suffices to keep the seeds $X_0, \ldots, X_{k-1}$ private, while $\alpha_1, \ldots, \alpha_k$ and p can be public. Note that in Section 8, we explain how to save the space of the secret key by speeding MRG. Because $p = 2^{31} - 1$ and $\alpha_1 = \alpha_2 = \ldots = \alpha_k < 2^{32}$ are popular values [38] for the modulo and parameter sizes, respectively, we make the following restrictions: $p < 2^{32}$ and $\alpha_1, \ldots, \alpha_k < 2^{32}$.

Following, we describe the steps followed by our block cipher, depicted in Figure 1.



Notations:
O_i=E(S_i, INFO$_i$); INFO$_i$=(Location,…);P_i=G(S_i); S_i =H(S_{i-1})=(H($\alpha_{0,i-1}$),…, H($\alpha_{k-1,i-1}$), p);

Figure 2. RFID tag sends (O_i, P_i) to RFID reader and renews its key using P_i.

Step 1–Pseudo-random Number Generation: We use the MRG in Equation (1) to generate the 4 byte pseudo-random numberX_k, as in Figure 1.

*Step 2–*The pseudo-random number generated this way is embedded in the plaintext by simply using addition modulo 256 byte-by-byte, same as in Reference [3]. We use 4 bytes for one plaintext packet P. Let $P = P_0P_1P_2P_3$ and $X_k = X_{k,0}X_{k,1}X_{k,2}X_{k,3}$ denote the 4 bytes of plaintext and random noise, respectively. The ciphertext $C = C_0C_1C_2C_3$ is therefore obtained by adding the pseudo-random number X_k to the plaintext P byte-by-byte and taking modulo 256: $P_i + X_{k,i} \mod 256 = C_i$, where $i = 1, 2, 3, 4$ For example, assume that the plaintext to be encrypted is *Stop*. In the ASCII decimal encoding, $S = 83$, $t = 116$, $o = 110$, and $p = 112$. Let X_k be the following base 16 number:$X_k = 12$ CF FF $4B_{16}$. Transformed in decimal, these numbers represent $12_{16} = 12$, $\mathrm{CF}_{16} = 207$, $\mathrm{FF}_{16} = 255$, $4B_{16} = 75$. Next, we obtain the ciphertext characters by addition modulo 256:

$$83 + 12 \bmod 256 = 95$$
$$116 + 207 \bmod 256 = 67$$
$$110 + 255 \bmod 256 = 109$$
$$112 + 75 \bmod 256 = 187$$

*Step 3–*This step constructs a permutation using the bytes ofX_k, like in Reference [3], with the purpose of making the uncovering of X_k very difficult. The permutation will be applied at the next step to the ciphertext C obtained at Step 2. X_k is a 4 byte pseudo-random number, $X_k = X_{k,0}X_{k,1}X_{k,2}X_{k,3}$. We define the permutation function $\Pi = \pi_0\pi_1\pi_2\pi_3$ as follows:

- $\pi_0 = X_{k,0} \bmod 4$;
- $\pi_i = n \bmod 4$, for $i = 1, \ldots, 3$ where n is the smallest integer larger or equal to $X_{k,i}$ s.t. $\pi_i \notin \{\pi_0, \ldots, \pi_{i-1}\}$.

*Step 4–*The ciphertext obtained in Step 2 is permuted using the permutation generated at Step 3. That is, the bytes of ciphertext C are scrambled such that the ith byte of Π (C) becomes the π_ith byte of C.

There are two major differences between our algorithm and the one in Reference [3]; first, the pseudo-random number generation formula is different: there are k initial seeds that need to be kept secret, instead of just one (in Section 7 we will show how these seeds can be easily generated), and second, the block cipher has a size of 4 bytes instead of 16 bytes. The subsequent steps of the algorithm are very similar: at Step 2 noise is added to the plaintext, while Steps 3 and 4 are concerned with the addition of a pseudo-random permutation in order to obtain security for the WSNs or RFID.

3.2. Privacy protection

As we know, one of the major drawbacks of RFID is that it has privacy issue, i.e., attackers can query RFID tags to obtain useful information, such as ID. Note that even though ID can be encrypted, if the encrypted ID is static, it is easy for attackers to track users against location privacy. In order to protect privacy, we present a hash chain-based scheme which uses our block cipher and show how it preserves anonymity for RFID, shown in Figure 2.

Each RFID tag has an initial information k, which is shared with the back-end database, and $s_1 = (\alpha_0, \ldots, \alpha_{k-1}, p) = H(s_0) = (H(\alpha_{0,0}), \quad H(\alpha_{k-1,0}), \quad p)$. The back-end database and the tag maintain a copy of the information pair $\langle \mathrm{ID}, s_1 \rangle$ initially.

In the ith transaction with the reader, the RFID tag

- sends answer (O_i, P_i) to the reader, where, $O_i = E(S_i, \mathrm{INFO}_i)$, $\mathrm{INFO}_i = (\mathrm{Location}, \ldots)$, $P_i = G(S_i)$, and $S_i = H(S_{i-1}) = (H(\alpha_{0,i-1}), H(\alpha_{k-1,i-1}), p)$, shown in Figure 2.
- renews key $s_{i+1} = H(s_i)$ as determined from previous key s_i.

where H and G are hash functions. The reader sends (O_i, P_i) to the back-end database. Since the information transmitted between reader and back-end database are assumed to be secured through a secure channel, the back-end database can receive (O_i, P_i) successfully.

Then the back-end database that received tag output (O_i, P_i) from the reader conducts the following operations:

(1) Calculates $P'_i = G(S_{i-1})$ for each S_1 in the maintained list, and checks if $P'_i = P_i$?
(2) If a match is found, then gets the corresponding ID and calculates $S_i = H(S_{i-1})$. The ID and calculated s_i will be used in the following authentication mechanism. Then updates the stored information pair $\langle \text{ID}, s_{i-1} \rangle$ to $\langle \text{ID}, s_i \rangle$. If a match is not found, the packets from tag will be dropped.
(3) Starts authentication process.

4. SECURITY ANALYSIS

We assess the security of our scheme by following the basic goals of confidentiality, authenticity, and privacy protection. With respect to confidentiality, we are concerned with concealing the contents of the data from an attacker. This process involves encryption and possibly randomization to help prevent multiple encryptions of the same plaintext from looking similar, i.e., semantic security. Integrity is concerned with being able to detect if a message has been tampered with while authenticity translates into making sure that the message was indeed sent by its intended sender.

4.1. Confidentiality

Each byte $X_{k,j}$, $j = 1, \ldots, 4$ of $X_k$is a number between 0 and 255. Therefore each byte of $X_k$can take 256 different values. The permutation Π is obtained through a many-to-one mapping. That is, there are $256^4/4! \approx 2^{32}/2^4 \approx 2^{28}$ different values that generate the same permutation byte π_i. Therefore, even if an attacker obtains the permutation function, finding the corresponding pseudo-random number that generates it is computationally infeasible.

We can further complicate the problem by using only half the bytes ($2^2 = 4$) in X_k to generate the permutation. Even if these bytes are recovered, this will not lead to revealing the value of X_k. According to the birthday attack, when $n = 4$, the smallest index number k for which $X_{k,i}$ mod 16 has more than 50% chance of colliding with one of the previously generated integers $\{\pi_0, \ldots, \pi_{i-1}\}$, is: $k \approx \sqrt{2n \ln\left(1/2\right)} - 1 \approx 1.35$

So starting from π_1, the value of π_i is more than 50% likely to be different from that of $X_{k,i}$.

4.2. Authenticity

Lets consider a replay attack on our cipher. In this type of attack, a third party can eavesdrop on messages sent between two nodes and then replay the messages at a future time. We present two approaches to prevent such an attack. The first approach and the most common one would be to use a counter and include it with every message. The drawbacks of this approach are the necessity of a counter synchronization protocol and the maintenance of a replay table containing the last value from every sender. The second approach consists in using a different key for the encryption of the next data message. This process introduces no extra overhead. However, it does not work very well in an environment with low packet delivery ratio. A solution to this problem would be to use rekeying with a reduced frequency.

Note that both approaches are also useful in achieving semantic security.

The permutation function used in Step 3 alters the order of the message's content, making it invulnerable to known plaintext attacks and to direct ciphertext analysis. In the context of a rekeying mechanism, in which a different encryption key is used with every transmission, chosen-plaintext attacks also do not apply.

4.3. Data aggregation

Let p_1, p_2 be two plaintext samples from our data set. An encryption transformation is additively homomorphic if:

$$p_1 + p_2 = D(E(p_1) + E(p_2)) \tag{4}$$

Here D denotes decryption, E denotes encryption, and $+$ signifies addition.

For our block cipher, property (4) stands if the same key is used for the encryption of the data to be aggregated.

In this case, data aggregation in WSNs by intermediate nodes is possible for messages sent between a source and destination. The messages need not be decrypted at every hop in order to achieve data aggregation. Instead, the encryption transformation allows for direct aggregation of the encrypted data. Moreover, in this case, secret key sharing among data aggregators is no longer necessary, saving space and resources.

Consider the following example. Suppose that p_1, p_2 translate to the base 10 numbers: $p_1 = 9 \quad 10 \quad 11 \quad 12$ and $p_2 = 83 \quad 116 \quad 110 \quad 112$. Also, consider that the encryption key, the same for both plaintext messages, is: $X = 12 \quad 207 \quad 255 \quad 75$. Then, the corresponding ciphertexts, obtained by addition modulo 256, are: $C_1 = 21 \quad 217 \quad 10 \quad 87$ and $C_2 = 95 \quad 67 \quad 109 \quad 187$, respectively. The permutation function is

$$\Pi = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}$$

After applying the permutation to C_1 and C_2, we obtain ciphertexts: $C'_1 = 21 \quad 87 \quad 217 \quad 10$ and $C'_2 = 95 \quad 187 \quad 67 \quad 109$, respectively. So far we have computed:

$$E(p_1) = C'_1 = 21 \quad 87 \quad 217 \quad 10 \quad \text{and}$$

$$E(p_2) = C'_2 = 95 \quad 187 \quad 67 \quad 109$$

The sum of the two obtained ciphertexts gives us:

$$E(p_1) + E(p_2) = 116 \quad 8 \quad 28 \quad 119$$

In order to decrypt the sum we need to first apply the inverse permutation $\Pi^{-1} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 2 & 3 & 1 \end{pmatrix}$ which gives us:

$$\Pi^{-1}(E(p_1) + E(p_2)) = 116 \quad 28 \quad 119 \quad 18$$

We extract the plaintext by doubling the values of the key and subtracting them byte by byte, which gives us:

$$(116 - 24) \bmod 256 = 92$$

$$(28 - 416) \bmod 256 = 126$$

$$(119 - 510) \bmod 256 = 121$$

$$(18 - 150) \bmod 256 = 124$$

For the left hand side of Equation (4), adding the plaintexts together gives us: $p_1 + p_2 = 92 \quad 126 \quad 121 \quad 124$, which proves that Equation (4) stands.

Finding the inverse permutation and then extracting the plaintext given the encryption key is straightforward.

4.4. Privacy protection

The proposed privacy protection scheme shown in Figure 2 achieves ID anonymity by never sending ID in plain or in some fixed form. Second, since the tag output is the result of one-way hash function, it keeps changing and is indistinguishable from truly random values. Thus the scheme avoids tag tracking. Third, we use the hash chain technique to renew the secret information contained in the tag, thus forward security is achieved.

5. ANALYSIS OF THE NUMBERS OF BASIC OPERATIONS OF OUR CIPHER

We assess the efficiency and power consumption of our block cipher by computing the number of basic operations that are being performed. This evaluation has two parts corresponding to generating the pseudo-random number (Section 3, Step 1) and the block cipher (Section 3, Steps 2, 3, and 4), respectively. We compute the number of basic operations for each part in the following sections.

5.1. Number of basic operations for generating one pseudo-random number

Based on Reference [2], we choose the coefficients $\alpha_i < 2^{30}$, $i = 1, \ldots, k$. The analysis below follows the one in Reference [3] very closely. Let $\alpha_{i,29} \ldots \alpha_{i,j} \ldots \alpha_{i,2}\alpha_{i,1}\alpha_{i,0}$ be the bit string representing α_i, $i = 1, \ldots, k$. We can write Equation (1) as

$$\begin{aligned}(\alpha_1 X_{k-1} + \ldots + \alpha_k X_0) \bmod p \\ = (\alpha_1 X_{k-1} \bmod p) + \ldots + (\alpha_{k-1} X_1 \bmod p) \\ + (\alpha_k X_1 \bmod p)\end{aligned}$$

In addition, as

$$\begin{aligned}\alpha_i X_{k-i} \bmod p = \left(\alpha_{i,29} 2^{29} X_{k-i} + \ldots + \alpha_{i,j} 2^j X_{k-i} + \ldots \right. \\ \left. + \alpha_{i,0} 2^0 X_{k-i}\right) \bmod p\end{aligned}$$

and

$$2^{j+1} X_{k-i} \bmod p = \left((2X_{k-i} \bmod p) \times 2^j\right) \bmod p$$

Based on these observations, we can compute the pseudo-random number X_k by the following algorithm, using only basic operations:

- There are $2k + 1$fetches.
- We consider $\left(\alpha_{k-i,j} = 1\right)$ in 50% of the cases, so line 9 is performed $k * 30/2$ times for each addition and each subtraction.
- Line 10 is performed $30 * k$ times for each shift and each subtraction.
- Line 12 is performed k times for each addition and each subtraction.
- We need one store for X_k.
- Total basic operations:
  - $(2k + 1) + 2 * k * 15 + 2 * k * 30 + 2 * k + 1 = 94k + 2$ basic operations.
  - If we consider $k = 1597$ like in Reference [2], then we need 150 120 basic operations to generate one pseudo-random number.
  - However, if we use $k = 47$ (another value for k suggested in Reference [2]), then the number of operations becomes 4420, which is a competitive value.

5.2. Number of basic operations for the MRG cipher

Our block cipher involves:

- One 32 bit XOR in Step 2.
- A Z_4 to Z_4 permutation construction and its application to 4 bytes. We need
  - Maximum 9 8-bit comparisons and 6 8-bit additions (for the while loop index) for construction the permutation. We consider an 8-bit comparison as an 8-bit XOR.
  - 4 8-bit fetches and 4 8-bit stores for applying the permutation.

1: input $X_0, X_1, ..., X_{k-1}, \alpha_1, \alpha_2, ..., \alpha_k, p$; // $2k+1$ fetch ops

2: $X_k \leftarrow 0$;

3: for $(i = 0 \text{ to } k-1)$

4: $X_{\alpha_{k-i,j}} \leftarrow 0$;

5: end for;

6: for $(i = 0 \text{ to } k-1)$

7: for $(j = 0 \text{ to } 29)$

8: if $(\alpha_{k-i,j} = 1)$

// 1add + 1subtr

9: $X_{\alpha_{k-i,j}} \leftarrow \text{mod}(X_{\alpha_{k-i,j}} + X_i, p)$;

//1 shift + 1 subtraction

10: $X_i \leftarrow \text{mod}(2X_i, p)$;

11: end for j ;

// 1add + 1subtr

12: $X_k \leftarrow \text{mod}(X_k + X_{\alpha_{k-i,j}}, p)$;

13: end for i ;

14: output X_k ; // 1 store

- There are $2k+1$ fetches.

- We consider $(\alpha_{k-i,j} = 1)$ in 50% of the cases, so line 9 is performed $k*30/2$ times for each addition and each subtraction.
- Line 10 is performed $30*k$ times for each shift and each subtraction.
- Line 12 is performed k times for each addition and each subtraction.
- We need one store for X_k.
- Total basic operations:
  - $(2k+1)+2*k*15+2*k*30+2*k+1=94k+2$ basic operations.
  - If we consider $k=1597$ like in [2], then we need 150120 basic ops. to generate one pseudo-random number.
  - However, if we use $k=47$, (another value for k suggested in [2]), then the number of ops becomes 4420, which is a competitive value.

6. PERFORMANCE ANALYSIS

Table I shows the number of basic operations necessary to construct the block cipher in Section 3. We analyze how many addition, XOR, shift, fetch, and store operations are needed on 128-bit, 32-bit, and 8-bit processors. It may also be of interest to see how the algorithm performs for packet sizes larger than 4 bytes. We present the overhead introduced when the packet size is 16 and 32 bytes on an 8-bit processor.

We can see that because our block cipher has a size of 4 bytes, no overhead is introduced when using a 32-bit processor as opposed to a 128-bit processor. For the 8-bit processor, 4-byte block, only the number of XORs is increased because one of the 10 XOR operations is performed on 32 bits, and it turns into eight XOR operations on an 8-bit processor. While the other operations stay the same. When the size of the packets is increased from 4 to 16 bytes (4 times larger) then the number of operations naturally becomes 4 times larger.

Table II summarizes the basic operations needed to generate one pseudo-random number, described in Section 3.1. The results are presented as a function of *k*, based on the MRG recursive formula (1). Again, the 32-bit processor introduces no overhead compared to the 128-bit processor, while on an 8-bit processor, eight times more operations need to be performed.

Table III shows the breakdown of the number of operations of the LCG based cipher from [3]. The numbers are mainly taken from the analysis in Reference [3] and represent the operations needed both for the block cipher and for generating one random number X_1 using the LCG formula (3).

We compare our results with the ones for the LCG-based cipher in Table IV. We can see that the size of k makes a huge difference in the efficiency of our scheme and that even

Table I. Number of basic operations in an MRG-based cipher.

Operation	128-bit processor (32-bit processor), 4 byte block	8-bit processor, 4 byte block	8-bit processor, 4 byte packet	8-bit processor, 16 byte packet	8-bit processor, 32 byte packet
Addition	0	0	0	0	0
XOR	$1+9=10$	$(4+4)+9=17$	17	68	136
Shift	0	0	0	0	0
Fetch	4	4	4	16	32
Store	4	4	4	16	32
Total	18	25	25	100	200

Table II. Number of basic operations for generating one 32-bit MRG pseudo-random number.

MRG operations	Number of MRG operations	Equivalent operations	128-bit processor, 4 byte block	32-bit processor, 4 byte block	8-bit processor, 4 byte block
32-bit addition	k	Addition	$30\,k+60\,k+2\,k=92\,k$	$92\,k$	$(92\times8)\,k=736\,k$
32-bit shift	0	Shift	$30\,k$	$30\,k$	$240\,kk$
32-bit fetch	$2\,k+1$	Fetch	$2\,k+1$	$2\,k+1$	$16\,k+8$
32-bit store	1	Store	1	1	8
32-bit multiplication	k−1				
32-bit moduli	1				
Total	$4\,k$		$124\,k+2$	$124\,k+2$	$992\,k+16$

Table III. Number of basic operations in an LCG-based cipher, plus the operations for generating 128-bits LCG pseudo-random numbers.

Operation	128-bit processor, 16 byte block	32-bit processor, 16 byte block	8-bit processor, 16 byte block	8-bit processor, 16 byte packet	8-bit processor, 32 byte packet
Addition	0	0	0	0	0
XOR	31	38	62	62	124
Shift	0	0	0	0	0
Fetch	31	38	62	62	124
Store	31	38	62	62	124
Total cipher operations	93	114	186	186	372
Total basic operations for an LCG pseudo-random number	197	1584	6304		
Total	290	1698	6490	6490	6676

moderate values for k, such as $k = 47$ are not practical.

The conclusion of our performance analysis is that k should be kept small in order to make our cipher efficient. The overhead introduced by using the MRG to generate pseudo-random numbers is essentially k times larger than that of using the LCG, when working with equal sized data sets.

7. SPECIAL CLASSES OF GENERATORS

The analysis above suggests that it may be of interest to search for more efficient types of MRGs which still lead to large periods, like the general MRG, but which also posses the advantage of having significantly fewer

Table IV. Number of basic operations in an LCG *vs.* an MRG-based cipher (Legends: P: processor BP = byte packet).

Cipher	128-bit P, 4 BP	32-bit P, 4 BP	8-bit P, 4BP	8-bit P, 16BP	8-bit P, 32 BP
LCG-cipher	290	1698	6490	6490	6676
MRG-cipher	$124k+2+18$	$124k+2+18$	$992k+16+25$	$992k+16+100$	$992k+16+200$
MRG-cipher K = 47	4438	4438	35385	35460	35560
MRG-cipher k = 6	584	584	4553	4628	4728

terms in the recursive formula. This means the overhead gets reduced while the large period property of MRGs is preserved.

We present below two different classes of MRGs that have these properties and analyze their advantages.

7.1. FMRG

Consider the MRG given by Equation (1). For the purposes of increasing the efficiency of the MRG we can use the idea stated in Reference [6], making as many of the α_i coefficients equal to 0. Specifically, if we choose$\alpha_1 = -1, \quad \alpha_i = 0 \quad \text{for} \quad 2 \le i \le k-1, \quad \text{and} \quad \alpha_k = B$, then we obtain a special form of MRG, called fast MRG (FMRG), as proposed in Reference [4]:

$$X_i = (BX_{i-k} - X_{i-1}) \bmod p, \quad i \ge k \tag{5}$$

This generator is very similar in form to the LCG in Equation (3), with the following observations:

- an increment is subtracted while in Equation (3) the increment is added;
- the FMRG increment is variable while in the LCG formula the increment is constant;
- because we are subtracting two positive integers, the FMRG has the advantage: it cannot produce an overflow. While the addition in the LCG can cause an overflow.

If in addition we want the FMRG to be portable, then a known good solution to the problem is to choose the coefficient B such that $B < \sqrt{p}$ [4,6]. Given $p = 2^{31} - 1$ and condition $B < \sqrt{p}$, several values have been found for B (please see Reference [4]), such that the MRG has maximum period $p^k - 1$. In particular, the value $B = 16807$ has been used extensively in empirical studies.

If we consider the computational complexity of the two ciphers, one based on the LCG, the other on the FMRG, obviously, there is no difference. The only difference is introduced by the fact that the size of the modulus is significantly smaller for the FMRG and so is the coefficient$B$, as compared to the corresponding coefficient a of the LCG.

However, if we consider the period of the two generators, the FMRG is significantly better.

For LCG: $x_i = 16807x_{i-1} \bmod (2^{31} - 1)$, the period is $2^{31} - 1 = 2147483646$.

For FMRG: $(k = 2)\, x_i = (39613x_{i-2} - x_{i-1}) \bmod (2^{31} - 1)$ the period is $\left(2^{31} - 1\right)^2 - 1 = 4611686014132420608$.

7.2. DX-k-s generators

It has been shown that MRGs display a lattice structure, that is, the consecutive k different values generated lie on a limited number $d$of equidistant parallel hyperplanes, instead of occupying k^d points placed uniformly in a d-dimensional cube. A necessary condition for a good lattice structure is that the sum of squares of the coefficients $\sum_{i=1}^{k} \alpha_i^2$ is large [12]. Consequently, one drawback of FMRGs is that they do not exhibit a good lattice structure.

Therefore, we present another special class of MRGs, called DX-k-s generators which extend FMRGs and are obtained by choosing s non-zero equal coefficients (instead of just 2 with FMRGs), equally spaced at $k/(s-1)$ distance while the rest of the coefficients are zeros [5].

The general form of the DX-k-s generators is

$$X_i = B\left(X_{i-k} + X_{i-\lfloor (s-2)k/(s-1) \rfloor} + \ldots + X_{i-\lfloor 2k/(s-1) \rfloor} + \right.$$
$$\left. + X_{i-\lfloor k/(s-1) \rfloor} + X_{i-1}\right) \bmod p, \quad i \ge k \tag{6}$$

It has been shown that an MRG with maximum period has the property of equi-distribution up to dimension k [13].

Paper [5] shows a complete factorization for $k = 120$, and presents values for B that lead to maximum periods. This way, the period has been extended to $p^{120} - 1 \approx 0.679 \cdot 10^{1120}$ and in addition DX-120 has the property of equi-distribution up to dimension 120. Values 1, 2, 3, and 4 are considered for s. The special case DX-120-4 generator is presented below: $X_i = B\,(X_{i-120} + X_{i-80} + X_{i-40} + X_{i-1}) \bmod p$.

As stated in Reference [5], the DX-120-4 generator is 10–15% less efficient than the DX-120-1 generator, but it exhibits better statistical properties, like a more suitable lattice structure.

Compared to an FMRG with $k = 4$, the DX-120-4 above also has four terms and necessitates the same number of basic operations, but it produces a $p^{120} - 1$ long period (compared to $p^4 - 1$ for the MRG) and therefore it constitutes a significant improvement while introducing no overhead.

Furthermore, values for B that lead to maximum periods are presented for DX-k-s generators, for $k = 643$ and $k = 1597$ and $s \in \{1, 2, 3, 4\}$ [2]. While generators: DX-643-4 $(k = 643,\ s = 4)$, $X_i = B(X_{i-643} + X_{i-428} + X_{i-214} + X_{i-1}) \bmod p$ and DX-1597-4 $(k = 1597,\ s = 4)$, $X_i = B(X_{i-1597} + X_{i-1064} + X_{i-532} + X_{i-1}) \bmod p$ introduce no overhead compared to the FMRG-4, they lead to maximum periods of approximately $10^{6000.4}$and $10^{14903.1}$, respectively. Table V summarizes the performance of these special types of generators.

8. SPEEDING THE MRG

In this section, we show some efficient ways to initialize the seed vector $X_0, \ldots, X_{k-1}$ and also mention proposed values for the coefficients which make the generator efficient and portable.

One approach is to use an LCG to generate the k seeds of the MRG [5]. We can start with an initial value of 1 and use the LCG with $a = 7^5$ and $m = 2^{31} - 1$, which easily

Table V. Number of basic operations in LCG and MRGs-based ciphers, *vs.* period length (Legends: P: processor BP = byte packet).

Cipher	Period length	128-bit P, 4 BP	32-bit P, 4 BP	8-bit P, 4 BP	8-bit P, 16 BP	8-bit P, 32 BP
LCG-cipher	$2^{128}-1$	$197+93=290$	$1584+114=1698$	$6304+186=6490$	$6304+186=6490$	$6304+372=6676$
LCG-cipher	$2^{31}-1$	$97+18$	$776+18$	$3104+25$	$3104+100$	$3104+200$
DX-1587-k		$(124k+2)+18$	$(124k+2)+18$	$(992k+16)+25$	$(992k+16)+100$	$(992k+16)+200$
FMRG-cipher	$\left(2^{31}-1\right)^2-1$	144	144	1033	1108	1208
DXk-4 cipher $k \in \{120, 643, 1597\}$	$\left(2^{31}-1\right)^k-1$	516	516	4009	4084	4184

generates a sequence of 1597 initial seeds, if we consider the DX-1597-4 generator.

Another way to easily seed the MRG is to start with a $4k$-byte number, which can be randomly generated, and assign the first 4 bytes to X_0, the next 4 bytes to X_1, and the last 4 bytes to X_{k-1}. Keep in mind that we are working with keys that are 32-bits long. Consider the DX-k-4 special-type MRG, which has proved the most efficient in our analysis. We therefore need one 16-bytes long number to start with and we can use it to seed our generator.

With respect to the coefficients, having many zeros is not desirable as a close to zero vector may lead to many subsequent generated vectors staying in the vicinity of zero, which detriments randomness. As shown in subsection 7.2, a good solution is to make all non-zero coefficients equal with possibly one exception [5]. In this case, only one value is stored, instead of k different values, and, in addition, only one multiplication needs to be performed (please see Equation (5)). Such suitable values leading to maximum periods have been found *via* computer generation and are presented in References [2,5].

9. RFID *VS.* WSN

An RFID tag is a small electronic device that serves as a unique identifier for an object. Most RFID applications are in the area of inventory tracking and management, automated logging, [16], as well as wireless telemedicine [8,10]. The main components are a transponder of RFID tag and a transceiver of RFID reader. The reader broadcasts a signal in a range usually up to 20 feet. When receiving the signal, tags can respond by sending information consisting for example in the ID number of the object. RFID tags are even more limited devices than WSNs in terms of cost, area, and power. As noted in References [11,16], the RFID tag ID numbers are short in length, usually, 8–16 bytes. Similar to WSNs, RFIDs are vulnerable to security attacks such as eavesdropping, unauthorized tag reading, and tracking. Since RFID scanners are very portable, the RFID tags can be read even after the object has left the supply chain without the knowledge of the individual carrying it. Moreover, an intruder could keep track of locations of an individual by repeatedly querying the item's tag.

Paper [11] proposes a lightweight block cipher for RFID systems. In their scheme, the 16-bytes size of the block cipher in Reference [3] is replaced with a more general $2L$-byte size. L can then be tuned so that the performance requirements are met while maintaining the security of the encryption at the same time.

Our block cipher is 4 bytes long, and so are our encryption keys. The extremely short length of our cipher makes it a suitable application for RFID communications. The size of the block cipher is determined by the size of the modulo, which in our case is $p = 2^{31} - 1$, thus in our scheme $L = 2$.

Clearly, as L increases, the security of the scheme increases. However, as stated in Reference [11], a large modulo size significantly contributes to the loss in

efficiency. In order to keep the key secret, it is more important to keep the parameters secret than to increase the modulo size. The strength of our idea lies in using an MRG to generate the encryption key. This allows for using small 4-byte numbers for the modulo and the generator parameters, that is, working with a four times smaller modulo size, while achieving an approximately 10^{5962} times larger period (see subsection 7.2).). The period length is a measure of the randomness of our key and therefore of the scheme's security. Basic RFID protocols which achieve mutual authentication between the reader and the tag [11] rely on the MRG to encrypt information sent between the tag and the reader. The efficiency of these protocols is thus directly affected by the efficiency of the MRG block cipher.

The privacy protection scheme in Section 3 is also suitable for RFID. The scheme can be used to achieve tag ID anonymity and forward security and to avoid tag tracking. The ID tag is never sent in plaintext and multiple encryptions of the same ID result in different ciphertexts, making impossible a direct ciphertext attack. This way semantic security is achieved and tag tracking is avoided. By using the hash chain technique, the secret information contained in the tag is permanently renewed, making the scheme invulnerable to chosen plaintext attacks and achieving security of the forwarded information.

10. CONCLUSIONS

We have constructed a block cipher based on the MRG. The MRG is being used for generating pseudo-random numbers with very large periods. The generated pseudo-random numbers are in turn used in our encryption scheme in order to add noise to the plaintext and are kept secret by the use of a pseudo-random permutation which works on the bytes of each pseudo-random number. We have compared our scheme with the one in Reference [3] and presented the results in terms of the number of basic operations of both ciphers. It is obvious that the computing time of an MRG is about k times longer than that of an LCG. However, the advantage of our scheme comes from using a very large period, which makes the generated numbers hard to uncover. Another advantage is given by the fact that by using a more complicated recursive formula, we can use smaller numbers for p and the coefficients $\alpha_1, \ldots, \alpha_k$. We can thus assume: $p < 2^{32}$ and $\alpha_1, \ldots, \alpha_k < 2^{32}$, while in Reference [3], the corresponding modulo$m$ and coefficient a are: $2^{63} < a < 2^{64}$ and $2^{128} < m < 2^{128}$. Therefore, our block cipher has smaller size of 4 bytes while the block cipher in Reference [3] has size 16 bytes.

Since our block size is only 4 bytes, our algorithm proves more effective if implemented on 32-bit processors, because there will be no increase in overhead in this case.

The choice of k is very important but we can see that even a small value like 6 can lead to a larger period than that of the LCG in Reference [3], while introducing less overhead.

Furthermore, it has been shown that by choosing most of the MRG's coefficients equal to 0, particular classes on MRGs, called FMRGs and DX-k-s generators can be obtained. These special generators are very similar in form and computational complexity to the LCGs, but in addition, they have the advantage of producing much larger periods.

We have found large periods, less or no extra overhead. Also, smaller numbers, less storage, memory, and larger periods enhanced security. Particularly important in WSNs and RFIDs where power consumption, memory and storage space are critical. They achieve enhanced security while consuming less resource!

ACKNOWLEDGEMENTS

This work was supported in part by the US National Science Foundation (NSF) under grants CNS-0716211, CNS-0737325, and CCF-0829827.

REFERENCES

1. Bellare M, Kilian J, Rogaway P. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences* 2000; **61**(3): 363–399.
2. Deng LY, Li H, Shiau JH, Tsai G. Design and Implementation of Efficient and Portable Multiple Recursive Generators with Few Zero Coefficients. Springer: Berlin, Heidelberg, 2006.
3. Foulley J-L. Multiple recursive random generators and their APL programmes, *Technical Report, INRA-SGQA*, 2005.
4. Lidl R, Niederreiter H. Introduction to Finite Fields and Their Applications, Cambridge University Press: Cambridge, UK, 1986.
5. Watson EJ. Primitive polynomials (mod 2). *Mathematics of Computation* 1962; **16**: 368–369.
6. Sun B, Xiao Y, Li C-C, Chen H, Yang TA. Security co-existence of wireless sensor networks and RFID for pervasive computing. *Computer Communications Journal* Special Issue on Secure Multi-Mode Systems and their Applications for Pervasive Computing, 2008; **31**(18): 4294–4303.
7. Hu F, Jiang M, Xiao Y. Low-cost wireless sensor networks for remote cardiac patients monitoring applications. *Journal of Wireless Communications and Mobile Computing* 2008; **8**(4): 513–529.
8. www.answers.com/topic/field-theory
9. www.answers.com/topic/simple-extension-1
10. www.answers.com/topic/field-extension
11. Xiao Y. Editorial. *International Journal of Security and Networks*, 2006; **1**(1/2): 1–1.
12. Shehab M, Bertino E, Ghafoor A. Workflow authorisation in mediator-free environments. *International Journal of Security and Networks* 2006; **1**(1/2): 2–12.

13. Jung EJ, Gouda MG. Vulnerability analysis of certificate graphs. *International Journal of Security and Networks* 2006; **1**(1/2): 13–23.
14. Kiayias A, Yung M. Secure scalable group signature with dynamic joins and separable authorities. *International Journal of Security and Networks* 2006; **1**(1/2): 24–45.
15. Franklin M. A survey of key evolving cryptosystems. *International Journal of Security and Networks* 2006; **1**(1/2): 46–53.
16. Hamadeh I, Kesidis G. A taxonomy of internet traceback. *International Journal of Security and Networks* 2006; **1**(1/2): 54–61.
17. Jhumka A, Freiling F, Fetzer C, Suri N. An approach to synthesise safe systems. *International Journal of Security and Networks* 2006; **1**(1/2): 62–74.
18. Evans JB, Wang W, Ewy BJ, Wireless networking security: open issues in trust, management, interoperation and measurement. *International Journal of Security and Networks* 2006; **1**(1/2): 84–94.
19. Englund H, Johansson T. Three ways to mount distinguishing attacks on irregularly clocked stream ciphers. *International Journal of Security and Networks* 2006; **1**(1/2): 95–102.
20. Zhu B, Jajodia S, Kankanhalli MS. Building trust in peer-to-peer systems: a review. *International Journal of Security and Networks* 2006; **1**(1/2): 103–112.
21. Ramkumar M Memon N. Secure collaborations over message boards. *International Journal of Security and Networks* 2006; **1**(1/2): 113–124.
22. Xiao Y, Jia X, Sun B, Du X. Editorial: security issues on sensor networks. *International Journal of Security and Networks* 2006; **1**(3/4): 125–126.
23. Wang H, Sheng B, Li Q. Elliptic curve cryptography-based access control. *International Journal of Security and Networks* 2006; **1**(3/4): 127–137.
24. Zheng J, Li J, Lee MJ, Michael A. A lightweight encryption and authentication scheme for wireless sensor networks. *International Journal of Security and Networks* 2006; **1**(3/4): 138–146.
25. Al-Karaki JN. Analysis of routing security-energy tradeoffs in wireless sensor networks. *International Journal of Security and Networks* 2006; **1**(3/4): 147–157.
26. Araz O, Qi H. Load-balanced key establishment methodologies in wireless sensor networks. *International Journal of Security and Networks* 2006; **1**(3/4): 158–166.
27. Deng J, Han R, Mishra S. Limiting DoS attacks during multihop data delivery in wireless sensor networks. *International Journal of Security and Networks* 2006; **1**(3/4): 167–178.
28. Hwu J-S, Hsu S-F, Lin Y-B, Chen R-J. End-to-end security mechanisms for SMS. *International Journal of Security and Networks* 2006; **1**(3/4): 177–183.
29. Wang X. The loop fallacy and deterministic serialisation in tracing intrusion connections through stepping stones. *International Journal of Security and Networks* 2006; **1**(3/4): 184–197.
30. Jiang Y, Lin C, Shi M, Shen XS. A self-encryption authentication protocol for teleconference services. *International Journal of Security and Networks* 2006; **1**(3/4): 198–205.
31. Owens SF, Levary RR. An adaptive expert system approach for intrusion detection. *International Journal of Security and Networks* 2006; **1**(3/4): 206–217.
32. Chen Y, Susilo W, Mu Y. Convertible identity-based anonymous designated ring signatures. *International Journal of Security and Networks* 2006; **1**(3/4): 218–225.
33. Teo JCM Tan CH, Ng JM. Low-power authenticated group key agreement for heterogeneous wireless networks. *International Journal of Security and Networks* 2006; **1**(3/4): 226–236.
34. Tan CH. A new signature scheme without random oracles. *International Journal of Security and Networks* 2006; **1**(3/4): 237–242.
35. Liu Y, Comaniciu C. Man H. Modelling misbehaviour in ad hoc networks: a game theoretic approach for intrusion detection. *International Journal of Security and Networks* 2006; **1**(3/4): 243–254.
36. Karyotis V, Papavassiliou S, Grammatikou M, Maglaris V. A novel framework for mobile attack strategy modelling and vulnerability analysis in wireless ad hoc networks. *International Journal of Security and Networks* 2006; **1**(3/4): 255–265.
37. Payne WH, Rabung JR, Bogyo T. Coding the lehmer pseudo number generator. *Communications of the Association for Computing Machinery* 1969; **12**: 85–86.
38. Deng LY. Efficient and portable multiple recursive generators of large order. *ACM Transactions on Modeling and Computer Simulation* 2005; **15**(1): 1–13.
39. Primitive polynomial. Available at: en.wikipedia.org/wiki/Primitive_polynomial [16 December 2007].
40. Deng LY, Lin DKJ. Random number generation for the new century. *American Statistician* 2000; **54**(2): 145–150.
41. Deng LY, Xu HQ. A system of high-dimensional, efficient, long-cycle and portable uniform random number generators. *ACM Transactions on Modeling and Computer Simulation* 2003; **13**(4): 299–309.
42. Sun B, Li C-C, Xiao Y. A lightweight secure solution for RFID, *Proceedings of IEEE GLOBECOM*, 2006.
43. Lei M, Xiao Y, Vrbsky SV, Li C-C, Liu L. A Virtual Password Scheme to Protect Passwords, *Proceedings of IEEE ICC*, 2008.
44. Hu F, Celentano L, Xiao Y. Error-resistant RFID-assisted wireless sensor networks for cardiac tele-healthcare.

Wireless Communications and Mobile Computing (WCMC) 2009; **9**(1): 85–101.
45. Hu F, Jiang M, Celentano L, Xiao Y. Robust medical *ad hoc* sensor networks (MASN) with wavelet-based ECG data mining. *Ad Hoc Networks* 2008; **6**(7): 986–1012.
46. Xiao Y, Shen X, Sun B, Cai L. Security and privacy in RFID and applications in telemedicine. *IEEE Communications Magazine*, Special issue on Quality Assurance and Devices in Telemedicine, 2006; 64–72.
47. Sun B, Li C, Wu K, Xiao Y. A lightweight secure protocol for wireless sensor networks. *Computer Communications Journal* Special Issue on Wireless Sensor Networks: Performance, Reliability, Security and Beyond, 2006; **29**(13–14): 2556–2568.
48. Knuth DE. The Art of Computer Programming, Vol. 2: Seminumerical Algorithms (3rd edn), Addison-Wesley, Reading, MA, 1988.
49. L'Ecuyer P. Bad lattice structures for vectors of non-successive values produced by some linear recurrences. *INFORMS Journal on Computing* 1997; **9**: 57–60.
50. Plumstead JP. (Boyar), Inferring a sequence generated by a linear congruence, *Proceedings of the 23rd Annual IEEE Symposium on the Foundations of Computer Science*, 1982; pp. 153–159.
51. Marsaglia G, Tsang WW, Wang J. Fast generation of discrete random variables. *Journal of Statistical Software* 2004; **11**(3): 1–11.
52. Xiao Y, Yu S, Wu K, Ni Q, Janecek C, Nordstad J. Radio frequency identification: technologies, applications, and research issues. *Journal of Wireless Communications and Mobile Computing* 2007; **7**(4): 457–472.

Authors' Biographies

Alina Olteanu received her B.S. degree in Computer Science and her M.S. degree in Applied Mathematics from the University of Bucharest and Polytechnic University of Bucharest, Romania in 2003 and 2005, respectively, and earned her Ph.D. degree in Computer Science from the University of Alabama, U.S.A. in 2009. Her research interests are in the areas of wireless network security, network performance optimization, and lightweight cryptography.

Dr Yang Xiao is currently with the Department of Computer Science at the University of Alabama. He currently serves as Editor-in-Chief for *International Journal of Security and Networks* (IJSN) and *International Journal of Sensor Networks* (IJSNet). He was the founder and Editor-in-Chief (2007–2009) for *International Journal of Telemedicine and Applications* (IJTA). His research areas are security, telemedicine, robot, sensor networks, and wireless networks. He has published more than 300 papers in major journals, refereed conference proceedings, and book chapters related to these research areas. He is a senior member of the IEEE. He was a voting member of the IEEE 802.11 Working Group from 2001 to 2004.

Dr Fei Hu is currently an associate professor in the Department of Electrical and Computer Engineering at the University of Alabama, Tuscaloosa, AL, U.S.A. His research interests are wireless networks, wireless security, and their applications in Bio-Medicine. His research has been supported by NSF, Cisco, Sprint, and other sources. He obtained his first Ph.D. degree at Shanghai Tongji University, China in Signal Processing (in 1999), and second Ph.D. degree at Clarkson University (New York State) in the field of Electrical and Computer Engineering (in 2002).

Bo Sun is an associate professor with the Department of Computer Science, Lamar University, Beaumont, Texas. His research interests include security issues of wireless networks and other communications systems. His research has been supported by the National Science Foundation and the 2006 Texas Advanced Research Program. He is a member of the IEEE.

Dr Hongmei Deng currently is a Lead Research Scientist at Intelligent Automation, Inc. (IAI), Maryland. Her primary research interests include wireless *ad hoc*/sensor networks, and wireless network security. Dr Deng received her Ph.D. in Electrical Engineering from University of Cincinnati in 2004, majoring in communications and computer networks. At IAI, she is currently leading several network and security related projects, such as secure routing in Airborne Networks, network service for Airborne Networks, DoS mitigation, agent-based intrusion detection system for MANET, acoustic sensor network for threat detection, and multi-sensor fusion.