A Lightweight Secure Solution for RFID

Bo Sun*
Dept. of Computer Science
Lamar University
Beaumont, TX 77710
bsun@cs.lamar.edu

Chung Chih Li
School of Information Technology
Illinois State University
Normal, IL 61761
cli2@ilstu.edu

Yang Xiao
Dept. of Computer Science
University of Alabama
Tuscaloosa, AL, 35487, USA
yangxiao@ieee.org

***Abstract*— Radio Frequency Identification (RFID) systems have provided promising solutions to effective identification of a large number of tagged objects. However, RFID systems suffer from unauthorized tag reading and potential eavesdropping, which becomes a challenging issue because of the shared radio medium and limited size and cost considerations in RFID. In this paper, based on a Linear Congruential Generator (LCG), we propose a lightweight block cipher that can meet the security and performance requirement of RFID systems. The trade-off between the security and overhead is discussed. Based on the proposed block cipher, we further propose a secure protocol for RFID that can provide data confidentiality and mutual authentication between the reader and the tag. We also provide performance analysis of our proposed block cipher.**

I. Introduction

In recent years, Radio Frequency Identification (RFID) have become more and more popular to provide automated identification systems in areas such as supply chain management, manufacturing, and inventory control [1]. Generally, RFID systems consist of the low-cost RFID transponders (also called *tags*), RFID readers, and a database (which stores records associated with tag contents). Readers broadcast an RF signal in a certain wireless range to access required information in the tags. Powered by the signal from the RFID reader (called *passive* tags) or an internal battery (called *active* tags), tags can respond to the reader by sending information such as the object identification data after receiving the reader's signal.

However, the universal deployment of RFID systems may pose different and widespread security and privacy challenges [2]. For example, Molnar *et al.* [4] identified that RFID suffers from the following potential types of security threats: tracking, hotlisting, and profiling. The intruder can *track* the location of the tag holder by periodically querying the tag. Also, without a proper security mechanism, the content of the tag data can also be released to an unauthorized adversary. The shared radio medium, the extreme scarce computational and storage capabilities make it challenging to design secure protocols to guard RFID against unauthorized tag reading and potential eavesdropping. These constraints make it impossible to deploy most of the traditional security primitives and protocols, such as the RSA [7], Diffie-Hellman algorithm [8], which usually incurs the storage of large keys and incurs heavy computation.

*This research was supported in part by the Texas Advanced Research Program under grant 003581-0006-2006.

There is always a good trade-off between complexity and security. In this paper, based on a Linear Congruential Generator (LCG) [6], we propose a lightweight block cipher that can meet the security and performance requirement of RFID systems. We are motivated by the fact that a good balance between security and efficiency can be achieved after properly arranging the numbers generated by the LCG. By adding random noise generated by a LCG and random permutations to RFID data, we demonstrate that our proposed cipher is secure enough for RFID. We further analyze that the security of our proposed cipher can be adjusted with the input length of the data. Armed with the lightweight block cipher, we demonstrate its usage in the mutual authentication between the reader and the tag. Security and performance of the proposed block cipher are also analyzed.

II. Threat Model, System Assumptions and Security Goals

We assume that an adversary can eavesdrop the traffic. Therefore, the adversary can perform cryptanalysis to deduce the secret. This raises the privacy issue of the tagged data. We assume that the message transmission between the reader and tag is one-hop. In this paper, when we design the secure protocol, we only consider *passive* tags, in which tags can only operate when the reader provides necessary energy. However, our proposed cipher is general and can be applied to *active* tags.

We consider a typical RFID system that consists of one RFID reader and multiple RFID tags. In order for our proposed mechanism to work, a secret key needs to be shared between the reader and the tag. Based on the different application and different types of tags (rewritable, write-once, etc), we can have different keying mechanism. Please see more details in Section IV-A.

We focus on the following security goals:

- *Confidentiality*: Many applications of RFID require secure tag readings so the sensitive data sent from the tag cannot be disclosed to attackers. For example, no customer wants the amount of money in a wallet to be easily determined by an unauthorized scanner. *Confidentiality* is typically achieved through encryption. The key point for RFID systems is how to accommodate the stringent resource limitation.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

- *Authenticity*: It ensures that data messages come from the intended sender. By achieving authenticity, we can prevent the intruder from scanning the data illegally.

III. LCG-BASED BLOCK CIPHER

We have already proposed a LCG-based secure protocol for Wireless Sensor Networks (WSNs) [5]. Based on the Plumstead's inference algorithm [9], we are motivated to embed the generated pseudo-random numbers with sensor data messages in order to provide security. Specifically, the security of our proposed cipher is achieved by adding random noise and random permutations to the original data messages. In this section, we briefly go over our LCG-based block cipher in the context of WSNs and explain why some parts are not applicable to RFID. Based on this, we further explain how we tailor the LCG-based cipher to RFID.

A. LCG

The simplest form of a Linear Congruential Generator (LCG) uses the following equation:

$$X_{n+1} = aX_n + b \mod m, \quad n = 0, 1, 2, \ldots \tag{1}$$

where a is the *multiplier*, b is the *increment*, and m is the *modulus*. X_n and X_{n+1} are the n^{th} and $(n+1)^{st}$ numbers, respectively, in the sequence generated by the LCG. X_0 is called the *seed* of the LCG. X_0, a, b, and m are the parameters of the LCG. The statistical properties of the pseudo-random numbers generated by a LCG depend on the selection of its parameter [10].

Based on this simplest LCG form and motivated by the idea that we can use the information itself to protect the random sequences, we pick up the Linear Congruential Generator (LCG) in its simplest form to produce pseudo-random numbers.

In addition to Plumstead's theoretical analysis, we implemented Plumstead's algorithm to observe how many pseudo-random numbers are actually needed to successfully recover the parameters of an unknown LCG, so we can adequately adjust our cipher to meet the security requirements.

We carried out experiments to demonstrate the empirical results of the Plumstead's algorithm. For $m \geq 2$ bytes, we used the Miller-Rabin Test, a very efficient randomized algorithm for primality tests, to select and determine prime numbers with an error rate less than $(\frac{1}{2})^{\lceil \log_2 m \rceil}$. Given m, we select 1000 sets of different parameters (a, b, m, and X_0). For each set of parameters, we generated the sequence of pseudo-random numbers $X_1, X_2, \ldots$.... We ran the *Plumstead*'s algorithm to decide how many X_i are needed to recover the set of parameters (a, b, m, and X_0).

The results of our experiments are shown in Table I, in which μ is the average number of samples needed to successfully infer the pseudo-random number sequence while δ is the standard deviation. The theoretical analysis of the Plumstead's algorithm is based on the worst case. In reality, however, the worst case rarely occurs. Experimental results show that the Plumstead's algorithm is much more powerful than what the theoretical analysis has suggested. We observe that the number of samples needed in average is far fewer than that of the worst case. Also, Table I contains the best case (min) and the worst case (max) for each size. The values of δ in Table I are standard deviations of the number of samples needed, which indicate that the worse case occurs rarely.

TABLE I
RESULTS OF PLUMSTEAD'S ALGORITHM

$\|m\|$ Bytes	μ	δ	min	max
1	5.438	0.939	5	12
2	5.617	1.221	5	17
4	5.554	1.082	5	15
8	5.586	1.114	5	16
16	5.802	1.764	5	31
32	6.105	3.149	5	57

Based on the results illustrated in Table I, we can see that the size of m does not prolong the inference process significantly. On the other hand, the size of m can significantly compromise the efficiency of the LCG. Therefore, for our application, instead of increasing the size of m, we need to hide the numbers generated. According to the results illustrated in Table I, *we meed to find a way to prevent the adversary from retrieving five or more consecutive numbers from the sequence, our cipher based on the LCG will be secure.* Our design follows the above principle by using the transmitted information to protect the sequence of random numbers and by using a re-keying mechanism.

In the context of WSNs, our proposed cipher encrypting a 16 Byte packet is illustrated in Fig. 1.

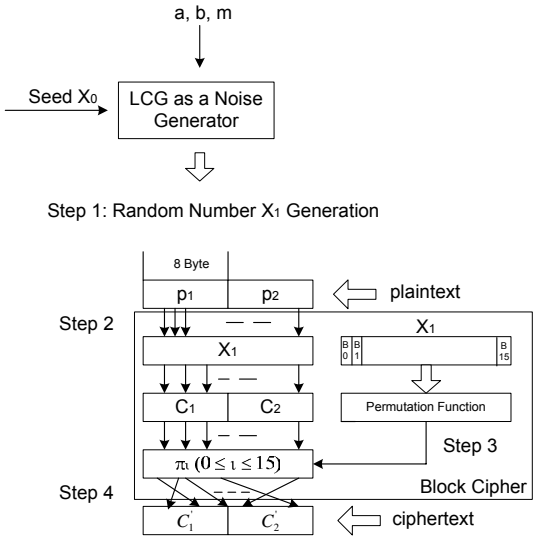

Fig. 1. Message Encryption of a 16 Byte Packet.

In the block cipher illustrated in Fig. 1, we first use the LCG to generate a random number X_1 (Step 1) and embed the pseudo-random number X_1 into the plaintext message (Step 2). We then apply the permutation function (Step 3). X_1 will also serve as the source of the permutation function. The final ciphertext is obtained after step 4. Please refer to [5]

for details.

Security analysis in [5] demonstrates that LCG based security mechanisms can satisfy the security requirement, i.e., confidentiality, authenticity, and integrity for WSNs.

IV. Applying LCG to RFID

The difference between WSNs and RFID makes it impossible to directly apply the block cipher in Fig. 1 to RFID. The main concern here is the computation capability. Tags are usually a miniature electronic circuit and contain between 500 and 5000 gates. For example, the ID number of tags is usually 64-128 bits (8 - 16 Bytes) in length. However, the input message length in Fig. 1 are 16 Bytes, which makes it unsuitable for short ID numbers. Note that simply padding 0s to the end of short ID to 16Bytes will only incur more computation overhead.

A. Keying mechanisms

For the read-only tags, a, b, m, and X_0 can be stored, and these numbers can be used to generate the random number for current usage.

For careful selection of a, b, m, and X_0 to maximize the security performance, please refer to [5]. We omit its discussion to save space.

In the very basic scheme, a secret X_1 is shared between the reader and the tag. Different approaches can be used to do this. For example, in the deployment of rewritable tags, X_1 can be a pseudo-random number and do not need to go through the LCG process. Also, X_1 can be stored in the database with the ID of the tag. For example, in a store, when the item is checked out and no tag is needed, the secrets can be erased from both the database and the tag. When a new item arrives in the store or the item is returned, we can let the powerful machines to generate random numbers, and write it into tags.

B. Length selection and Performance analysis

Based on this consideration, we tailor Fig. 1 to a more general and lightweight block cipher, as illustrated in Fig. 2. In Fig. 2, the length of the input message and X_1 is $2L$ Bytes. The permutation function $\pi_0\pi_1 \ldots \pi_{2L-1}$ is obtained based on X_1, i.e., each π_i is determined by the first $\lceil \log_2 L \rceil + 1$ bits of B_i and $\pi_0, \pi_1, \ldots \pi_{i-1}$.

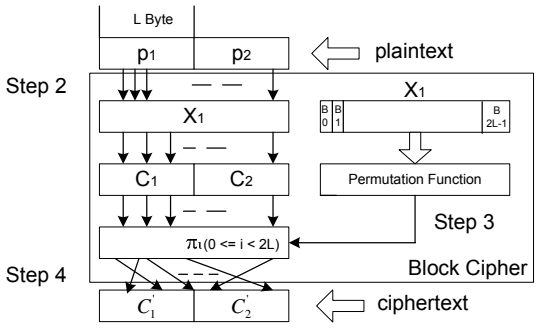

Fig. 2. LCG based block cipher for RFID.

1) Security Analysis: The mapping from X_1 to the permutation is many-to-one. Under the chosen-plaintext attack, the adversary may successfully obtain the permutation function if he is allowed to choose and encrypt $2L$ plaintexts. However, the same permutation function may be constructed based on $\frac{256^{2L}}{(2L)!}$ many different pseudo-random numbers X_1. When $L = 4$, for example, $\frac{256^{2L}}{(2L)!} \approx 2^{49}$, which is not feasible to guess the correct X_1. Using Stirling's approximation for $(2L)!$, one can see that increasing L with $L < 128$ will also increase $\frac{256^{2L}}{(2L)!}$. Thus, a larger L within the reasonable range for applications will lead to better security. Nevertheless, this will also increase computational overhead. L can be treated as a security parameter to the system.

When $L = 4$, we use the first three bits ($8 = 2^3$) in B_i to construct the permutation function. The probability that the values in B_i do not introduce collisions, according to Birthday [6] attack, when $n = 8$ and $k \approx \sqrt{n \ln 0.5^{-1}} - 1 \approx 2.62$. The probability of $B_k \mod 2L \in \{\pi_0, \pi_1, \ldots, \pi_{k-1}\}$ (the probability of collision) is at least 0.5. Therefore, starting from π_2, the value of π_i is not likely to be the value of $B_i \mod 2L$. As i becomes larger, the chance of collisions becomes larger and the chance that the attacker obtains the right value for B_i becomes smaller.

C. Discussion

Our cipher is designed based on the following belief: while the pseudo-random numbers are generated to protect the message, the entropy of the message itself can in turn protect the pseudo-random numbers. Thus, if the message sent out from the tags is almost flat, i.e., with very low entropy, our encryption in Step 2 alone is insecure since too many random bits can be recovered and, consequently, the size of the possible key space will be largely reduced. For this reason, we introduce the permutation in our cipher in Steps 3 and 4 to guarantee that even if our cipher is applied to a low entropy environment, the security of our cipher will not be significantly compromised.

Moreover, the encryption in Step 2 alone cannot resist *known-plaintext attack* in case the message in a sequence of transmitted packages is known to the adversary. To fix this problem, the permutation function takes on in Step 3, in which the random numbers generated by the LCG play an extra role in altering the original order of the content of the message.

So far, we are not aware of any known-plaintext attack against our proposed block cipher. Even a plaintext-ciphertext pair is given, there is no easy way to separate the two factors, noise and permutations, involved in the ciphertext. Likewise, a direct ciphertext analysis does not seem possible.

We can see that with the increase of L, the security of the proposed block cipher is increased. Therefore, for an RFID system, we can tune L to provide a good trade-off between security and performance.

D. Mutual Authentication

A very basic mutual authentication protocol in RFID is presented in Fig. 3. We can see that the efficiency of the

protocol depends on the efficiency of the LCG block cipher.

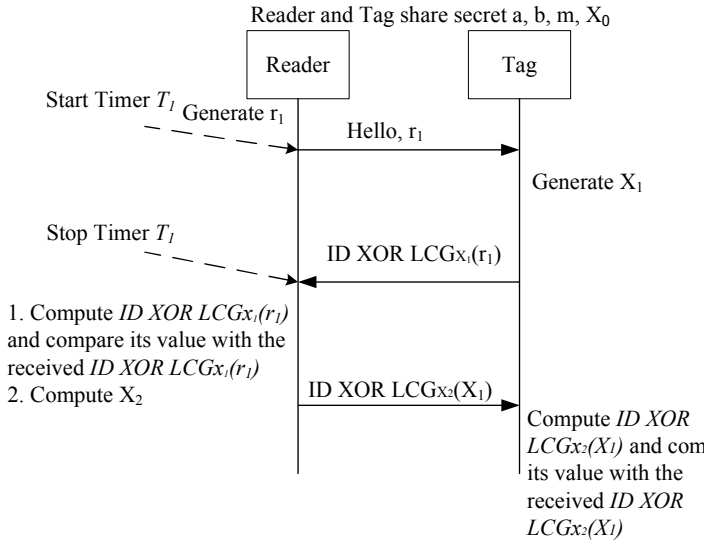

Fig. 3. Mutual Authentication between the Reader and the Tag.

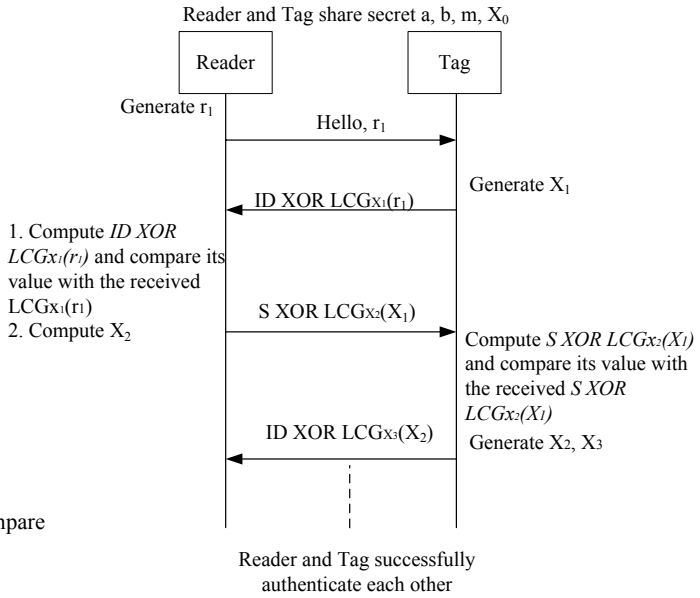

Fig. 4. Mutual Authentication between the Reader and the Tag for Binary Tree Walking Scheme.

In Fig. 3, the reader and the tag share the secret a, b, and m. To read the data from the tag, the reader generates a random number r_1 and starts a timer T_1. r_1, with a length of $2L$ Bytes, is sent to the tag. The purpose of T_1 is to prevent the potential intruder from adequate time to decrypt X_1. The time period that T_1 will expire depends on many factors, including the length of X_1. After the tag receives the challenge r_1, it uses the encryption scheme illustrated in Fig. 2 to encrypt r_1, denoted as $LCG_{X_1}(r_1)$. To provide its identification, the tag sends $ID\ XOR\ LCG_{X_1}(r_1)$ to the reader.

To authenticate the reader, along with the encrypted r_1, a new challenge number r_2 is needed to be sent from the tag to the reader. In our case, to reduce the overhead in the tag side, we use X_1 as the challenge, instead of r_2. Therefore, after the reader receives the message from the tag, a new message in the format $ID\ XOR\ LCG_{X_2}(X_1)$ is sent from the reader to the tag.

E. Collision Avoidance ID Authentication

In practice, there may exist collisions when the reader wants to find a specific tag [1]. A *binary tree walking* scheme can be used to resolve the collisions. In this case, there exist multiple rounds of communications between the reader and tags.

Let n denote the number of tags (leaves) in a binary tree. A node of depth d is labeled with a binary string S of length d, and has two children with depth $d+1$: the left child is labeled $S||0$ and the right child is labeled $S||1$. The binary tree walking algorithm is a recursive depth-first search for the reader to find all IDs of tags [1].

In this case, each tag (edge in the binary tree) is associated with a secret and this secret is shared with the reader. We have the protocol illustrated in Fig. 4. In Fig. 4, we omit the timer to make the figure better illustrated.

Similar to Fig. 3, the reader initiates to poll the tags by generating a random number r_1. When the reader queries a node with binary string S, all tags whose IDs have S as the prefix respond to the next bit. Each tag in the left subtree of the node sends 0, and each tag in the right subtree of the node sends 1. The reply from each tag can be used by the reader to authenticate the tag. Also, the reader queries the tag with the binary string S. This message can be used by the tag to authenticate the reader. The mutual authentication can go to the next level after it succeeds at the current level. If the reader passes all secrets in the path, the reader is authenticated.

Please note that in the tag's side, only a sequence of $X_1, X_2, \ldots, X_n$ are needed. These Xs, in turn, can be used by the tag to authenticate the reader.

We briefly analyze the complexity of the protocol illustrated in Fig. 4. Given L, assume that the computation overhead incurred by applying the LCG one time is O. Let's consider a balanced binary tree with n leaves. Here we only focus on the tag side because they represent the performance bottleneck. The protocol illustrated in Fig. 4 requires $\lceil \lg n \rceil$ invocations of the LCG. Therefore, the computation overhead for the tag to authenticate itself to the reader is $O * \lceil \lg n \rceil$. We detail the analysis of O in Section V.

V. Performance Analysis

The overhead is determined by the *Number of Basic Operations* our block cipher and protocol incur. We consider Addition, XOR, Shift (1 bit), Fetch (fetch a value from the main memory to a register), and Store (store a value in a register to the main memory) as our basic operations. To make our comparison plausible, we consider the cost of performing one general n-bits multiplication as $\frac{n}{2}$ additions and $\frac{n}{2}$ shifts in average on n-bit registers. Since a division can be reduced to

a multiplication, we use the same estimation for the division. Also, the same estimation is made to the general modulo.

We have some special cases: a multiplication by 2 is a left-shift operation; the operation of $(n \ mod \ 32)$ is considered one XOR operation (in fact, we need a bitwise AND). We consider that n basic operations on a 32-bit-processor are equivalent to $8n$ basic operations on a 8-bit processor. This is because each operation will be broken into 4 operations plus 4 store operations. This may be oversimplified since some necessary bookkeeping such as handling the carry bits may be required, but we ignore them for simplicity.

In Table II, the first column is the name of the basic operations. The second, third, and fourth columns list the number of basic operations needed for a $2L$ bytes block, where L is 4, 8, and 16 on a 8-bit processor, respectively. Here *Op.* is the abbreviation for *Operation*. Please note that the number of operations presented in Fig. II does not include the operations to generate random numbers.

Operation	8-bit Op. L=4	8-bit Op. L=8	8-bit Op. L=16
Addition	0	0	0
XOR	31	62	124
Shift	0	0	0
Fetch	31	62	124
Store	31	62	124
Total	93	186	372

TABLE II

NUMBERS OF BASIC OPERATIONS IN LCG-BASED CIPHER.

Table III briefly analyzes the number of basic operations to generate a pseudo-random number. Here we use $L = 8$. In Table III, the first column illustrates the basic LCG operations involved in the random number generation. The second column illustrates the number of corresponding LCG operations. The third column lists the corresponding number of the basic operations for a 16 byte block on a 128-bit processor. The fourth column lists the corresponding number of the basic operations for a 16 byte block on an 8-bit processor.

LCG Operations	Number of LCG Op	Equivalent Op	8-bit Op. 2L byte
2L Byte Addition	1	Addition	4128
2L Byte Shift	0	Shift	2048
2L Byte Fetch	4	Fetch	128
2L Byte Store	1	Store	32
2L Byte Multiplication	1		
2L Byte Moduli	1		
Total	8		6304

TABLE III

NUMBERS OF BASIC OPERATIONS FOR GENERATING ONE LCG PSEUDO-RANDOM NUMBER WHEN L=8.

VI. RELATED WORK

Xiao *et al.* [1] discussed the security and privacy issues of RFID, and their solutions in telemedicine. Juels *et al.* [2] proposed the use of "selective blocking" by "blocker tags" to protect consumers from unwanted RFID tag scanning. Molnar *et al.* [4] suggested a novel security architecture for library RFID. Inoue *et al.* [11] proposed two approaches to protect the privacy of RFID. In one approach, the permanent ID is concealed under a private ID that users give. In the other approach, partial ID sequence is assigned to an object and the rest is given by user-assignable RFID tags. Weis *et al.* [12] suggested several security enhancement mechanisms for RFID. Vajda *et al.* [3] proposed a set of extremely lightweight tag authentication protocols for RFID systems. Compared to his approaches, we provide stricter security and performance analysis of our proposed cipher and secure protocol.

VII. CONCLUSIONS AND FUTURE WORK

We propose a LCG-based lightweight block cipher that can meet the security requirements of RFID. The security of our proposed cipher is achieved by adding random noise and random permutations to the original data messages. We further analyze the security performance of our proposed cipher. Based on the proposed block cipher, we present protocols for the mutual authentication between the reader and the tag.

The extreme scarce resource in the tags of RFID makes it challenging to provide suitable security mechanisms. We plan to further tailor our block cipher based on the specific characteristics of tags, and propose different protocols to enhance RFID security and privacy.

REFERENCES

[1] Y. Xiao, X. Shen, B. Sun, and L. Cai, "Security and Privacy in RFID and Applications in Telemedicine", *IEEE Comm. Mag.*, Apr., 2006, pp. 64-72.

[2] A. Juels, R. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy", *ACM CCS'03*, Washington D.C., 2003, pp. 103-111.

[3] I. Vajda and L. Buttyan, "Lightweight Authentication Protocols for Low-Cost RFID Tags", 2nd Workshop Security in Ubiquitous Comp., 2003.

[4] D. Molnar, and D. Wagner, "Privacy and Security in Library RFID Issues, Practices, and Architectures", *ACM CCS'04*, Washington DC, 2004, pp. 210-219.

[5] B. Sun, C.-C Li, K. Wu, and Y. Xiao, "A Lightweight Secure Protocol for Wireless Sensor Networks", *E*lsevier Computer Communications journal special issue on Wireless Sensor Networks: Performance, Reliability, Security, and Beyond, 2005.

[6] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 2nd Edition, John Wiley & Sons, 1996.

[7] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM*, 21(2):120-126, 1978.

[8] W. Diffie and M. E. Hellman, "New Directions in Cryptography", *IEEE Transaction on Information Theory*, IT-22:644C654, November 1976.

[9] J.P. Plumstead (Boyar), "Inferring a Sequence Generated by a Linear Congruence," *23rd Annual IEEE Symposium on the Foundations of Computer Science*, pp. 153-159, 1982.

[10] D.E. Knuth, "Deciphering a Linear Congruential Encryption," *IEEE Tran. on Information Theory*, vol. 31, no. 1, pp. 49-52, Jan. 1985.

[11] S. Inoue and H. Yasuura, "RFID Privacy Using User-Controllable Uniqueness," *RFID Privacy Workshop*, MIT, MA, USA, Nov. 2003.

[12] S.A. Weis, S.E. Sarma, R. L. Rivest, and D. W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," *Security in Pervasive Computing*, vol. 2802 of LNCS, pp. 201-212, 2004.