Security Co-existence of Wireless Sensor Networks and RFID

Bo Sun[a], Yang Xiao[b], Chung Chih Li[c], and T. Andrew Yang[d]

[a] Department of Computer Science
Lamar University, Beaumont, TX USA 77710

[b] Department of Computer Science
The University of Alabama, Tuscaloosa, AL USA 35487

[c] School of Information Technology
Illinois State University, Normal, IL USA 61761

[d] Department of Computer Science
University of Houston - Clear Lake, Houston, TX USA 77058

Email: bsun@cs.lamar.edu, yangxiao@ieee.org, cli2@ilstu.edu, yang@uhcl.edu

Abstract

Recent advances in wireless networks and embedded systems have created a new class of pervasive systems such as Wireless Sensor Networks (WSNs) and Radio Frequency IDentification (RFID) systems. WSNs and RFID systems have provided promising solutions for a wide variety of applications. However, security and privacy concerns have raised serious challenges on these systems. These concerns have become more apparent when WSNs and RFID systems co-exist. In this article, we first briefly introduce WSNs and RFID systems. We then present their security concerns and related solutions. Finally, we propose a Linear Congruential Generator (LCG) based lightweight block cipher that can meet security co-existence requirements of WSNs and RFID systems.

1 Introduction

Recent advances in wireless networks and embedded systems have created a new class of pervasive systems such as Wireless Sensor Networks (WSNs) and Radio Frequency IDentification (RFID) systems. WSNs and RFID have made a variety of new and exciting applications viable. For example, WSNs have been used in areas such as health monitoring, scientific data collection, environmental monitoring, and military operations. RFID systems have become more and more popular to provide automatic identification systems in areas such as supply chain management, payment systems, manufacturing, and inventory control [1]. The integration of WSNs and RFID systems has also opened up new opportunities in areas such as healthcare systems and wireless telemedicine.

WSNs usually comprise a large number of inexpensive, small, and battery-powered sensor nodes. One representative sensor node is Berkeley-designed MICA2 Motes. Equipped with wireless communication modules and microcontrollers, each sensor node can monitor physical or environmental conditions, such as temperature, light, acoustic, etc., and collaborate to transmit data to a base station. WSNs are usually resource-constrained on processing power, memory, bandwidth, and energy consumption. For example, powered by AA batteries, MICA2 Motes consist of an 8 MHz 8-bit Atmel ATMEGA128L CPU with only 4KB RAM for data, 128KB program memory, 512KB flash memory, and 38.4kbps data rate ratio.

RFID systems usually consist of simple and low-cost RFID *tags*, more powerful RFID *readers*, and a database which stores records associated with tag contents. Generally, a reader broadcasts an RF signal within a certain wireless range to access digital data stored in tags. Powered by a signal from an RFID reader or an internal battery, tags can respond to the reader by replying with information such as object identification data. Because tags are usually manufactured on a massive scale and any additional circuitry in tag design may incur extra cost, tags should be kept as lightweight as possible. For example, one tag in the form of Electronic Product Codes (EPC) may only contain 128-512 bits of read-only storage, 32-128 bits of volatile read-write memory, and $1,000$-$10,000$ gates [5].

Unfortunately, the wide deployment of these low-cost devices is often subject to various kinds of attacks and thus raises serious security and privacy concerns. For example, WSNs are often deployed in untrusted or hostile environment such as battlefield to perform mission-critical tasks, in which an adversary can eavesdrop traffic, inject malicious messages, replay old messages, and so on. The pervasive deployment of tags makes RFID systems suffer from security threats such as tracking, hotlisting, and profiling [3], which render tag data susceptible to an unauthorized reader and allow an adversary to gather private information illegally. The extreme resource-constrained nature of tags also makes it possible for attackers to insert a forgery or counterfeiting tag into an RFID system without being detected. All these vulnerabilities indicate that WSNs and RFID systems are not readily to be deployed for security-sensitive tasks without first addressing their security problems. Moreover, with the emergence of exciting applications such as wireless telemedicine, the co-existence of WSNs and RFID systems poses even more challenges for suitable security mechanisms.

In this article, we first briefly introduce WSNs and RFID systems. We then present their related security and privacy issues, as well as related solutions. We demonstrate that existing security solutions do not consider co-existence issues of WSNs and RFID systems. Finally, we propose a Linear Congruential Generator (LCG) based lightweight block cipher that can meet security co-existence requirements. Note that this article does not address integration issues (such as network architecture, networking protocols, etc.) of WSN and RFID systems. Instead, we aim at providing co-existent security solutions for such systems, i.e., we consider consistency and integration of security protocols.

2 Wireless Sensor Networks

One WSN may be composed of hundreds or thousands of miniature sensor nodes, or motes, which are fitted with an on-board processor. The low-cost battery-powered sensor nodes have extremely limited energy supply, stringent processing and communications capabilities, and scarce memory.

Sensor nodes are usually densely deployed in a sensor field in order to continuously monitor surrounding areas. In a sensor application, each sensor has the capability to collect data such as temperature, humidity, light condition, and so on, depending on targeted applications. After sensor nodes collect data, they can locally carry out some simple computations, and collaboratively route data to a base station for analysis. A base station may be a fixed node or a mobile node capable of connecting WSNs to a communications infrastructure (for example, the Internet) where users can have access to reported data. In order to reduce the amount of raw data transmitted to a base station and to save energy, sensor nodes often need to perform aggregation operations so that only processed information, for instance, the mean, max, or min of sensed raw data, is transmitted.

2.1 Security and Privacy Issues and Solutions for WSNs

The lack of physical security combined with unattended operations make sensor nodes prone to a high risk of being captured and compromised. The wireless broadcast nature may result in privacy breaches of sensitive information during data transmission. Therefore, security and privacy issues of WSNs have attracted a lot of research efforts. In the following, we list a brief taxonomy of WSN attacks and their representative solutions.

2.1.1 Attacks

- *Physical Attacks*: Sensor nodes may be left unattended for a long time. Therefore, attackers may have a high chance to compromise WSN nodes. From the hardware perspective, attackers can gain complete access to microcontrollers in sensor nodes and thus obtain sensitive information stored in node memory. From the software perspective, TinyOS, the most widely used Operating System in WSNs, and various applications may also suffer from well-know exploitations such as buffer overflow. All these enable attackers to extract relevant secrets, and insert malicious data to the network very easily.

- *Attacks at Physical Layer*: *Jamming* is one of the most important attacks at physical layer. Aiming at interfering with normal operations, an attacker may continuously transmit radio signals on a wireless channel. Equipped with a powerful node, an attacker can send high-energy signals in order to effectively block wireless medium and to prevent sensor nodes from communicating. This can lead to Denial-of-Service (DoS) attacks at the physical layers.

- *Attacks at Link Layer*: The functionality of link layer protocols, such as those specified in 802.15.4/ZigBee standards, is to coordinate neighboring nodes to access shared wireless channels and to provide link abstraction to upper layers. Attackers can deliberately violate predefined protocol behaviors at link layer. For example, attackers may induce collisions by disrupting a packet, cause exhaustion of nodes' battery by repeated retransmissions, or cause unfairness by abusing a cooperative MAC-layer priority scheme [6]. All these can lead to DoS attacks at the link layers.

- *Attacks at Network Layer*: In WSNs, attacks at routing layer may take many forms. For example, routing control packets exchanged among sensor nodes can be spoofed, replayed, or altered. In this way, routing logic can be compromised. Data packets may also be selectively dropped, replayed, or modified by compromised nodes. Besides these, WSNs also suffer from wormhole and sinkhole attacks, in which messages may be lured or tunneled to a particular area through compromised nodes. Attackers may also launch Sybil attack. Therefore, a single node may present multiple identities to other nodes in a network.

- *Attacks targeting at WSN Services and Applications*: In this respect, we use *localization*, *aggregation*, and *time synchronization* as examples.

  Accurate *locations* play a critical role in many WSN applications. For example, location information can be used in geographic routing protocols to facilitate sensor nodes to make routing decisions based on their own and their neighbors' locations. To enable location discovery protocols, WSNs are equipped with *beacon* nodes, which often know their own locations and can transmit *location references* to other sensor nodes that do not have location information. Location references contain locations of beacon nodes. Based on received known locations and features of received signals, other sensor nodes can then apply various algorithms to estimate their locations. Basically there are two types of localization protocols: *range-based* and *range-free*. In *range-based* protocols, absolute point-to-point distance or angle estimates can be applied to calculate location. *Range-free* protocols have no such assumptions. Unfortunately, most of the proposed localization schemes become target of attacks. For example, an adversary may compromise a beacon node to provide incorrect location references, replay beacon packets previously intercepted at other locations, or manipulate beacon signals to provide incorrect beacon signals. Therefore, sensor nodes may be misled to derive totally wrong locations. This results in a significant negative impact on relevant applications.

  Aggregation has been proven to be an important primitive to reduce communication overhead and to save energy for WSNs. The aggregation node can collect raw data from a subset of sensor nodes and aggregate (for example, average, sum, min, max) the received raw data and transmit them out toward a base station. However, an adversary can easily compromise one or more aggregation nodes and thus insert bogus readings or nonexistent events into the networks.

  By synchronizing local clocks through message exchanges, time synchronization protocols are also critical for WSNs to support many WSN applications such as tracking and localization. Unfortunately, like many other services, time synchronization protocols also suffer from many kinds of attacks [8]. These attacks try to violate time synchronization protocols from many different aspects. For example, in *masquerader* attacks, an attacker A may pretend to be a different node B and exchange malicious information with node C, leading C to derive wrong information. In *message manipulation* attacks, an adversary may drop, modify, or even forge exchanged timing messages.

2.1.2 Defense Mechanism

We summarize representative countermeasures for above mentioned attacks. These countermeasures aim at protecting the integrity, authenticity, and confidentiality of WSNs.

- *Key Management and Trust Setup*: One research problem is how to set up secret keys and bootstrap secure communications among sensor nodes in WSNs. To do so, a wide variety of *key management* schemes have been proposed. The first approach is based on *trusted-server* scheme, in which a trusted server is responsible for key aggrement among nodes. However, because a trusted server is not a suitable assumption for WSNs, this approach is not desirable. The second type of approaches is *public-key* based schemes, in which asymmetric cryptography is used. However, because sensor nodes are often resource-constrained, this type of approach is not suitable either. The third type of approaches is based on *key-predistribution* schemes, where key information is distributed among all nodes prior to deployment. *Key-predistribution* schemes seem most appropriate for WSNs. Therefore, we list several representative approaches in the following.

  Eschenauer *et al.* propose a random key-predistribution scheme, in which each sensor node receives a random subset of keys (called *key rings*) from a key pool before deployment. Relying on probabilistic key sharing, two sensor nodes can find one single common key within a key ring to act as a shared key secret. Based on Eschenauer's scheme, there are more research work with further security enhancement and more security analysis. For example, Chan *et al.* propose a "q-composite" scheme, in which q common keys are needed, instead of just one. Therefore, Chan's scheme increases the resilience of WSNs against node capture. Chan's scheme needs the same amount of key storage, while requiring attackers to compromise many more nodes. With a little more computation overhead and without using too much additional memory, Du *et al.* further propose to improve network resilience based on Eschenauer's scheme. Zhu *et al.* propose Localized Encryption and Authentication Protocol (LEAP), in which sensor nodes are preloaded with initial keys, from which further keys can be established to set up different keys for future usage. Utilizing deployment knowledge which may be available *a priori*, Du *et al.* also propose a random key-distribution scheme which can guarantee that any two neighboring nodes can find a common secret key with a certain probability [9] [10]. With the recent progress of sensor platforms, there is an emerging trend to demonstrate that public key cryptography, such as RSA and Elliptic Curve Cryptography (ECC), may be feasible for WSN related security applications. With optimized implementation of RSA and ECC, it now becomes reasonable to run public key techniques on popular sensor nodes. This makes public key based key management schemes a desirable candidate for WSNs.

- *Secrecy and Authentication*: Based on established keys, these are various kinds of authentication and privacy mechanisms in WSNs. For example, TinySec [7], a software based lightweight encryption mechanism, offers a feasible and efficient security solution for WSNs at the link layer. In this option, using a block cipher based on Skipjack, each packet is encrypted and appended a Message Authentication Code (MAC) to achieve message integrity and confidentiality.

  In WSNs, an end-to-end encryption scheme is usually impractical. Instead, trust can be set up between neighboring nodes and a hop-by-hop encryption can then be performed. For example, with the help of symmetric cryptography techniques, an Interleaved Hop-by-Hop Authentication (I-LHAP) [11] scheme is proposed to detect and to filter out injected false data in WSNs. In I-LHAP, MACs are jointly generated by a group of nodes for a sensing target. A message is attached with multiple MACs and each MAC is generated using one group key. Because a node usually only knows one group key, it is very difficult for one node to modify a message without being detected. Based on a Linear Congruential Generator (LCG), Sun *et al.* [2] propose a new block cipher that is suitable for constructing a lightweight secure protocol for resource-constrained wireless sensor networks.

- *Secure Aggregation*: Most existing secure aggregation schemes employ cryptographic techniques. Przydatek *et al.* propose Secure Information Aggregation (SIA) to defend against *stealthy* attack, whose purpose is to make a user accept false aggregation results. In SIA, aggregators need to prove and commit in order to illustrate that answers provided by these aggregators are good approximations of true values. Chan *et al.* [12] further propose a secure hierarchical in-network aggregation scheme, which can limit an adversary's ability to manipulate aggregation results. In this way, an adversary can gain no additional influence over final aggregation results through manipulation. The scheme of Yang *et al.* [13] divides an aggregation tree into subtrees, each of which reports aggregation results to a base station. The base station then identifies suspicious reports and each suspected group needs to prove the correctness of reported aggregates.

- *Secure Localization*: Different secure localization schemes have also been explored. Liu *et al.* [14] propose two techniques to survive malicious attacks against location discovery. The first approach is derived from the "consistency" among received beacon signals. Based on the observation that malicious location references are usually inconsistent with benign ones, a Minimum Mean Square Estimation (MMSE) based approach is applied to examine the inconsistency among received location references and to filter out malicious location references. In the second approach, a deployment field is divided into a grid of cells. Based on received location references, each node may "vote" on the locations at which this node may reside. After processing all of the received location references, the cell with the highest number of votes is the estimated location. In [15], Du *et al.* present a scheme by letting sensor nodes verify whether derived locations are consistent with deployment knowledge to identify location anomalies.

3 Radio Frequency Identification System

Envisioned as a replacement for barcodes, billions of RFID tags have been deployed on the market for various applications. For example, pharmaceutical companies have embedded RFID chips in drug containers to track the theft of highly controlled drugs. Airline companies may use RFID tags to track and route passenger bags. In all these applications, a tag is attached to a physical object and contains a digital number associated with that object. RFID *readers* broadcast a radio signal which contains an identifier in order to locate the object. Based on different operating frequencies (for example, 13.56 MHz or 915 MHz), RFID systems may have different reading ranges (for example, 1m or 3m). Because many RFID tags may be in the range of a reader at the same time, collisions may happen. Collision-avoidance protocols, such as *binary tree walking* protocol [1], are thus proposed to resolve this collision.

The pervasive nature of RFID systems make stored data increasingly distributed among different parties. This raises many new privacy and security for RFID systems. Because a reader is little more than a radio transceiver, it is thus relatively easy for attackers to obtain illegitimate readers and to query RFID tags for sensitive information. For example, consumer products labeled with insecure tags may reveal private information when queried by unauthorized readers. Many RFID protocols have no explicit authentication procedures. This may result in serious privacy concerns.

3.1 Security and Privacy Concerns for RFID

Because identifiers of RFID tags may be static and never change, this facilitates *tracking* attacks - to enable an attacker to track the movement of products. An adversary can also *hotlist* important objects, based on which activities of targeted objects can be *profiled* [3]. RFID systems also suffer from *tag spoofing* and *cloning*, in which an adversary can physically access tags or use an unauthorized reader to read tags in order for spoofing. This allows an adversary to clone targeted tags.

3.2 Security and Privacy Solutions for RFID

Tags lack necessary computational, communication, storage, and power resources to support strong cryptographic authentication schemes. These limitations make securing RFID systems a very challenging task.

So far, efficient and low-cost authentication represents one of the most important security efforts for RFID systems. Molnar *et al.* [3] suggest a scheme to achieve mutual authentication between a tag and a reader. The scheme requires a shared secret s between a tag and a reader. The basic idea is to let both a reader and a tag generate a random number r_1 and r_2, respectively. To begin with, the reader sends r_1 to the tag. The tag then sends $(r_2, \sigma = ID \oplus f_s(0, r_1, r_2))$ to the reader, where f_s is a keyed pseudo-random function. This message enables the reader to authenticate the tag. In order for the tag to authenticate the reader, the reader needs to send a message $r = ID \oplus f_s(1, r_1, r_2)$ to the tag.

To enhance security, Dimitriou [4] uses a secure one-way hash function and random session identifiers. In this way, tag responses may remain untraceable. After a reader sends a *nounce* N_R to a tag, the tag sends $(h(ID_i), N_T, h_{ID_i}(N_T, N_R))$, where N_T is the *nounce* generated at the tag. After using this message to authenticate the tag, the reader can send $h_{ID_{i+1}}(N_T, N_R)$, based on which the tag can authenticate the reader.

Observing that human beings and tags bear similarities such as limited computing resources shared by both parties, Juels *et al.* [5] propose a new and efficient authentication protocol HB^+, which is improved based on human authentication protocol *Hopper and Blum* (HB). In HB^+, a reader and a tag share two random secret x and y. The tag also needs to generate a random factor b. Each time a reader sends a query to a tag, the reader sends a new challenge $a \in \{0,1\}^k$. Based on a, b, x, and y, the tag generates z and sends z to the reader. The reader verifies z before accepting the tag as legitimate.

4 Linear Congruential Generator based Approach

Prevention-based approaches are still the most widely studied approaches to provide security mechanisms for WSNs and RFID systems. Resource-constrained nature of small devices presses a need for lightweight primitives to provide security solutions. In this section, based on a Linear Congruential Generator (LCG), we propose a lightweight block cipher that can meet the security and performance requirement of WSNs and RFID systems.

Based on the Plumstead's inference algorithm [2], we are motivated to embed the generated pseudo-random numbers with messages in order to provide security. Specifically, the security of our proposed cipher is achieved by adding random noise and random permutations to original data messages.

4.1 LCG Basics

The simplest form of an LCG uses the following equation:

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

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 an LCG depend on the selection of its parameter.

Starting with this simplest LCG 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 implement the 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 security requirements.

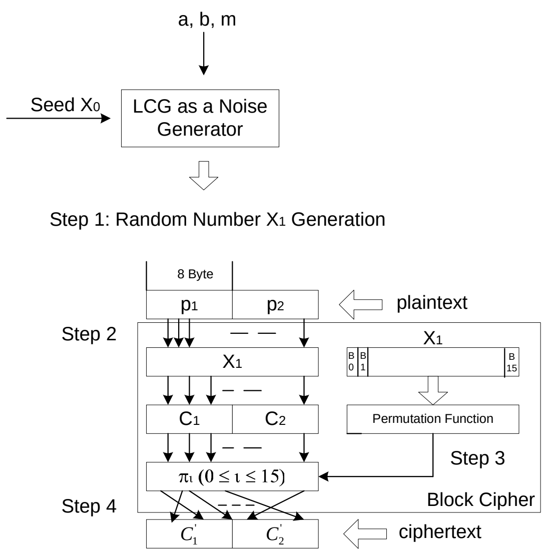
We carry out experiments to measure the impact of m on the security performance of the LCG. We test the module, m, from 1 byte and double its size up to 32 bytes. For $m \geq 2$ bytes, we used the Miller-Rabin Test, a very efficient randomized algorithm for primality tests, to select and determine prime numbers with an error rate less than $(\frac{1}{2})^{\lceil \log_2 m \rceil}$. Given m, we select $1,000$ sets of different parameters (a, b, m, and X_0). For each set of parameters, we generate the sequence of pseudo-random numbers $X_1, X_2, \ldots, X_n$. We run 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 1, 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 1 contains the best case (min) and the worst case (max) for each size. The values of δ in Table 1 indicate that the worse case occurs rarely.

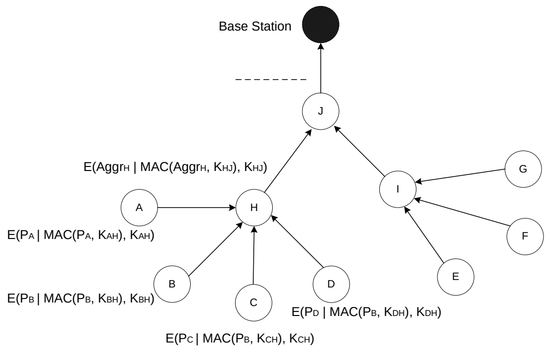
Table 1: Results of Plumstead's Algorithm

$\|m\|$ 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 1, we can see that the size of m does not prolong the inference process significantly. This is because, from the theoretical point of view, the size of m does not affect the number of internal states. Therefore, for an LCG, instead of increasing the size of m, we need to hide the numbers generated. Also, from the results illustrated in Table 1, we can see that *if we can 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.*



(a) Message Encryption of a 16 Byte Packet.


Base Station
J
E(AggrH | MAC(AggrH, KHJ), KHJ)
G
I
A
H
F
E(PA | MAC(PA, KAH), KAH)
E
B
D
C
E(PB | MAC(PB, KBH), KBH)
E(PD | MAC(PB, KDH), KDH)
E(PC | MAC(PB, KCH), KCH)

(b) Hop By Hop Security Protocol.

Figure 1: LCG based Hop by Hop Security Protocol for WSNs.

4.2 LCG based Security Protocols in WSNs

In this section, we briefly introduce our LCG based security protocols for WSNs. Please refer to [2] for more details.

Our proposed cipher to encrypt a 16 Byte packet is illustrated in Fig. 1(a). In Fig. 1(a), 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.

Based on an LCG based block cipher, the overall hop-by-hop security scheme is illustrated in Fig. 1(b). In Fig. 1(b), sensor nodes, such as nodes A, B, C, and D have monitored some events and transferred the readings to their immediate aggregator, node H. Each sensor node appends a MAC to the plaintext message P and uses their shared secret keys with H to encrypt the whole message. After H receives the readings, H uses the corresponding secret to decrypt and to authenticate the received messages. This time, node H appends a new MAC to the aggregated result and uses its shared secrets with its immediate aggregator, node J, to encrypt the whole message. The process continues until the result reaches the base station.

4.3 LCG based Security Protocols for RFID

4.3.1 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.

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 an item is returned, we can let the powerful machines to generate random numbers, and write it into tags.

4.3.2 Length Selection and Security Analysis

Based on this consideration, we tailor Fig. 1(a) 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 $\pi_0, \pi_1, \ldots \pi_{i-1}$.

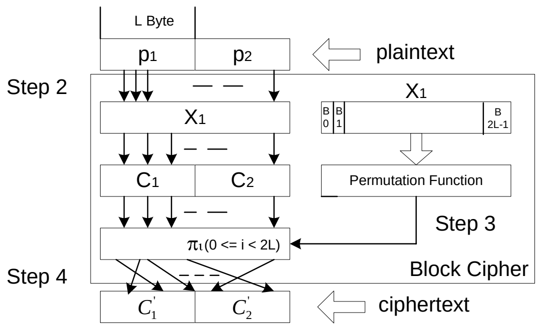


Figure 2: LCG based Block Cipher for RFID.

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.

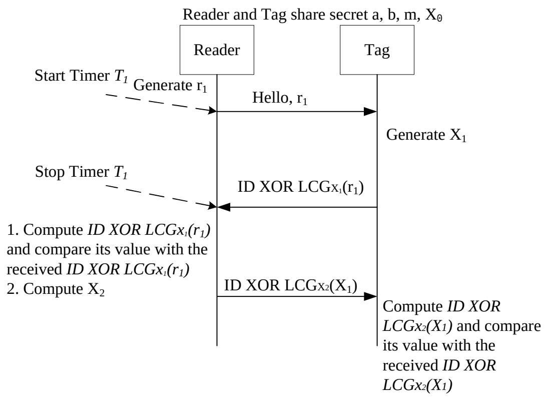
When $L = 4$, we use the first three bits ($8 = 2^3$) in B_i to construct the permutation function. The probability that the values in B_i do not introduce collisions, according to Birthday attack, when $n = 8$ and $k \approx \sqrt{n \ln 0.5^{-1}} - 1 \approx 2.62$. The probability of $B_k \mod 2L \in \{\pi_0, \pi_1, \ldots, \pi_{k-1}\}$ (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.

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 packets 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.

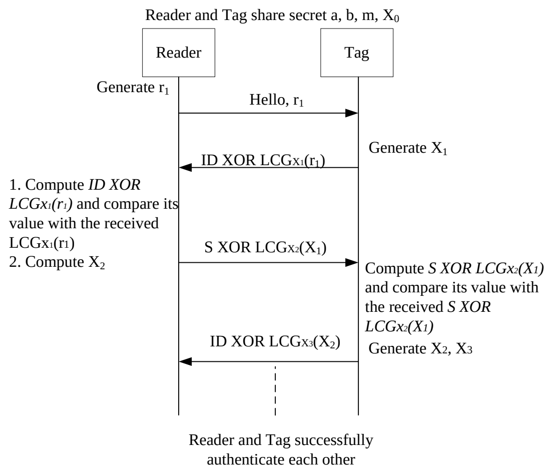
So far, we are not aware of any *known-plaintext attack* against our proposed block cipher. Even a plaintext-ciphertext pair is given, there is no easy way to separate the two factors, noise and permutations, involved in the ciphertext. Likewise, a direct ciphertext analysis does not seem possible.

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. Moreover, except the length of plaintext messages, the block cipher illustrated in Fig. 1(a) is the same as that illustrated in Fig. 2. This can facilitate security integration of WSNs and RFID systems.



Reader and Tag share secret a, b, m, X0
Reader
Tag
Start Timer T1
Generate r1
Hello, r1
Generate X1
Stop Timer T1
ID XOR LCGX1(r1)
1. Compute ID XOR LCGX1(r1) and compare its value with the received ID XOR LCGX1(r1)
2. Compute X2
ID XOR LCGX2(X1)
Compute ID XOR LCGX2(X1) and compare its value with the received ID XOR LCGX2(X1)

(a) Mutual Authentication between a Reader and a Tag.



Reader and Tag share secret a, b, m, X0
Reader
Tag
Generate r1
Hello, r1
Generate X1
ID XOR LCGX1(r1)
1. Compute ID XOR LCGX1(r1) and compare its value with the received LCGX1(r1)
2. Compute X2
S XOR LCGX2(X1)
Compute S XOR LCGX2(X1) and compare its value with the received S XOR LCGX2(X1)
ID XOR LCGX3(X2)
Generate X2, X3
Reader and Tag successfully authenticate each other

(b) Mutual Authentication between a Reader and a Tag for Binary Tree Walking Scheme.

Figure 3: LCG based Security Protocol for RFID.

4.3.3 Mutual Authentication

A very basic mutual authentication protocol in RFID is presented in Fig. 3(a). In Fig. 3(a), a reader and a tag share secret a, b, X_0, and m. To read 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 potential intruders from decrypting X_1 given adequate time. After the tag receives the challenge r_1, it uses the LCG based encryption scheme 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 r_2 is needed to be sent from the tag to the reader. In our case, to reduce the overhead on 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 of (ID XOR $LCG_{X_2}(X_1)$) is sent from the reader to the tag.

4.3.4 Collision Avoidance ID Authentication

In practice, there may exist collisions when a reader wants to find a specific tag [1]. A *binary tree walking* scheme can be used to resolve the collisions. In this case, there may exist multiple rounds of communications between a 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. 3(b). In Fig. 3(b), we omit the timer to make the figure better illustrated.

Similar to Fig. 3(a), a reader initiates to poll 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. Note that on 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.

4.3.5 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 2, 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. 2 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 2: Numbers of Basic Operations in LCG-based Cipher.

Table 3 briefly analyzes the number of basic operations to generate a pseudo-random number. Here we use $L = 8$. In Table 3, 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 equivalent operations on an 8-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 3: Numbers of Basic Operations for Generating One LCG Pseudo-Random Number when L=8.

5 Conclusions

In this article, we first briefly introduce WSNs and RFID systems. We then present their privacy and security concerns and related solutions. We finally propose a Linear Congruential Generator (LCG) based lightweight block cipher that can meet security co-existence requirements of WSNs and RFID systems.

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] B. Sun, C.-C Li, K. Wu, and Y. Xiao, “A Lightweight Secure Protocol for Wireless Sensor Networks,” *Elsevier Computer Communications Journal Special Issue on Wireless Sensor Networks: Performance, Reliability, Security, and Beyond*, 2006, pp. 2556-2568.

[3] D. Molnar and D. Wagner, “Privacy and Security in Library RFID: Issues, Practices, and Architectures,” *ACM CCS’04*, Washington, DC, USA, Oct. 2004, pp. 210-219.

[4] T. Dimitriou, “A Lightweight RFID Protocol to Protect Against Traceability and Cloning Attacks,” *IEEE SecureComm’05*, Athens, Greece, Sept. 2005, pp. 59-66.

[5] Stephen A. Weis, “New Foundations for Efficient Authentication, Commutative Cryptography, and Private Disjointness Testing,” Ph.D. Dissertation, MIT, May 2006.

[6] Y. Xiao, H.-H Chen, B. Sun, R. Wang, and S. Sethi, “MAC Security and Security Overhead Analysis in the IEEE 802.15.4 Wireless Sensor Networks,” *EURASIP Journal on Wireless Communications and Networking*, Article ID 93830, 2006.

[7] C. Karlof, N. Sastry, and D. Wagner, “TinySec: A Link Layer Security Architecture for Wireless Sensor Networks,” *ACM Sensys’04*, Baltimore, MD, 2004, pp. 162-175.

[8] H. Song, S. Zhu, and G. Cao, “Attack-Resilient Time Synchronization for Wireless Sensor Networks,” *Elsevier Journal of Ad Hoc Networks Special Issue on Security in Ad hoc and Sensor Networks*, 5(1), 2007, pp. 112-125.

[9] W. Du, J. Deng, Y.-S Han, P. Varshney, J. Katz, and A. Khalili, “A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks,” *ACM Transactions on Information and System Security (TISSEC)*, Vol. 8, No. 2, May 2005, pp. 228-258.

[10] W. Du, J. Deng, Y.-S Han, and P. Varshney, “A Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge,” *IEEE Transactions on Dependable and Secure Computing*, Vol. 3, No. 2, 2006, pp. 62-77.

[11] S. Zhu, S. Setia, S. Jajodia, and P. Ning, “An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data Injection in Sensor Networks,” *IEEE Symposium on Security and Privacy*, Oakland, CA, 2004, pp. 260-272.

[12] H. Chan, A. Perrig, and D. Song, “Secure Hierarchical In-Network Aggregation in Sensor Networks,” *ACM CCS’06*, Alexandria, VA, 2006, pp. 278-287.

[13] Y. Yang, X. Wang, S. Zhu, and G. Cao, “SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks,” *ACM MOBIHOC’06*, Florence, Italy, May 2006, pp. 356-367.

[14] D. Liu, P. Ning, and W. Du, “Attack-Resistant Location Estimation in Sensor Networks,” *IEEE IPSN ’05*, Los Angeles, CA, April 2005, pp. 99-106.

[15] W. Du, L. Fang, and P. Ning, “LAD: Localization Anomaly Detection for Wireless Sensor Networks,” *Journal of Parallel and Distributed Computing (JPDC)*, Vol. 66, No. 7, July 2006, pp. 874-886.

[16] B. Sun. C.-C, Li, and Y. Xiao, “A Lightweight Secure Solution for RFID,” *Proc. of IEEE GLOBECOM’06*, Nov. 2006, San Francisco, CA.