Computer Communications 31 (2008) 4294–4303

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Security co-existence of wireless sensor networks and RFID for pervasive computing ☆

Bo Sun [a], Yang Xiao [b], Chung Chih Li [c], Hsiao-Hwa Chen [d,*], T. Andrew Yang [e]

[a] Department of Computer Science, Lamar University, USA
[b] Department of Computer Science, The University of Alabama, USA
[c] School of Information Technology, Illinois State University, Normal, USA
[d] Department of Engineering Science, National Cheng Kung University, Tainan City, Taiwan
[e] Division of Computing and Mathematics, University of Houston – Clear Lake, USA

ARTICLE INFO

Article history:
Available online 7 June 2008

Keywords:
RFID
Wireless sensor network
Security
Pervasive computing

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 provide promising solutions for a wide variety of applications, particularly in pervasive computing. 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 for pervasive computing.

Published by Elsevier B.V.

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, particularly for pervasive computing. 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 the 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 A2 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 2 AA batteries, MICA2 Motes consist of an 8 MHz 8-bit Atmel ATMEGA128L CPU with only 4 KB RAM for data, 128 KB program memory, 512 KB flash memory, and 38.4 kbps 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 1000–10,000 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

☆ This research was supported in part by the Texas Advanced Research Program under Grants 003581-0006-2006 and 011711-0005-2006 and the US National Science Foundation (NSF) under Grants DUE-0633445, CNS-0716211, CNS-0737325, and DUE-0633469.

* Corresponding author. Tel.: +886 6 2757575x63320; fax: +886 6 2766549.
E-mail addresses: bsun@my.lamar.edu (B. Sun), yangxiao@ieee.org (Y. Xiao), cli2@ilstu.edu (C.C. Li), hshwchen@ieee.org (H.-H. Chen), yang@uhcl.edu (T. A. Yang).

0140-3664/$ - see front matter Published by Elsevier B.V.
doi:10.1016/j.comcom.2008.05.035

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 their security and privacy issues and related solutions in Section 2. We then briefly discuss RFID systems and their security issues and related solutions in Section 3. We demonstrate that existing security solutions do not consider co-existence issues of WSNs and RFID systems. In Section 4, we propose a Linear Congruential Generator (LCG) based lightweight block cipher that can meet security co-existence requirements. Based on LCG, we also present suitable security protocols for WSNs and RFID systems and analyze their performance in Section 4. It should be noted that this article does not intend to 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 the security protocols for WSNs and RFID systems.

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. One example of sensor networks is illustrated in Fig. 1.

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 [18], 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.

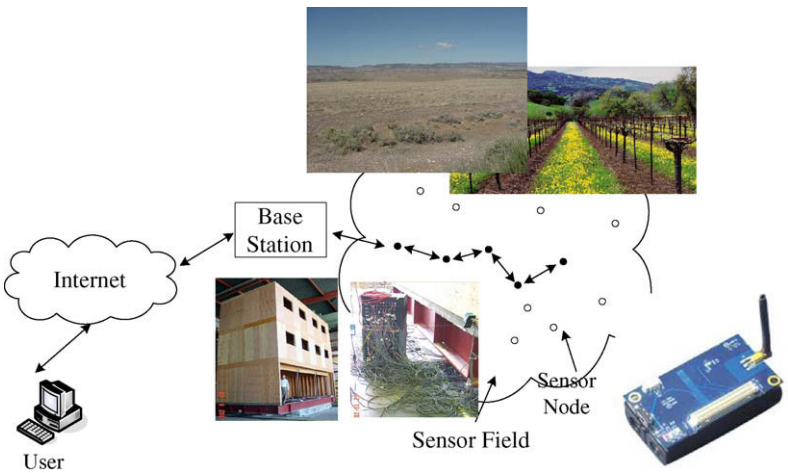

Fig. 1. One example of wireless sensor networks.

- *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 and aggregation 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.

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 agreement 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. Liu *et al.* [11] also propose schemes to detect misused keys in 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) [12] 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. [13] 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. [14] 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. [15] 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 [16], 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.

An RFID system usually consists of RFID tags and RFID readers. A tag is attached to a physical object and contains a digital number associated with that object. Tags usually have very low cost, limited storage, and extremely limited computing capability. Tags may be powered by readers wirelessly (called *passive* tags) or by a battery (called *active* tags). RFID readers are devices that read/interrogate tags, and each reader is equipped with antennas, a transceiver, and a processor. The reader broadcasts 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, 1 m or 3 m). Because many RFID tags may be in the range of a reader at the same time, collisions may happen. Collision-avoidance protocols are thus proposed to resolve this collision.

Binary tree walking protocol [1] is one such protocol. Binary tree walking protocol is a recursive depth-first search for the reader to find all tag IDs. When the reader queries a node with a binary string S of length d, all tags whose IDs have S as the prefix response 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. If their next bits are different, a collision happens, and the reader sequentially runs the algorithm on the node with the label $S\|0$ and the node with the label $S\|1$. If there is no collision and all tags send the same bit a, the reader will sequentially run the algorithm on the node with the label $S\|a$, ignoring the other child node. If the algorithm reaches a leaf, it outputs its N-bit ID. In this way, IDs of all tags are output.

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. In [22], Song et al. propose a new authentication protocol for RFID that can resist tag information leakage, tag location tracking, replay attacks, and denial of service attacks.

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 sing 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.

It is easy for us to think of linear algorithms when *efficiency* and *simplicity* come to our top priorities. Motivated by the fact that we can use the information itself to protect the random sequences, we can use the linear pseudo-random number generators (PRNGs) as an efficient mechanism to protect the data transmission. Motivated by this, we pick up the LCG in its simplest form to produce pseudo-random numbers. The reason we

select the LCG is because it is the simplest, most efficient, and a well-studied pseudo-random number generator.

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:

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

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 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 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.

4.2. Plumstead's algorithm

Assume Eq. (1) is a LCG with the fixed parameters a, b, m, and X_0, where $m > \max(a, b, X_0)$. The algorithm will find a congruence $X_{n+1} = \hat{a}X_n + \hat{b} \bmod m$, possibly with a different multiplier and increment but generating the same sequence as the fixed congruence does. The inference consists of two stages as follows. Let $Y_i = X_{i+1} - X_i$.

- **Stage I:** In this stage, we find $\hat{a}$ and $\hat{b}$ as follows:
  1. Find the least t such that $d = gcd(Y_0, Y_1, \ldots Y_t)$ and d divides Y_{t+1}.
  2. For each i with $0 \leqslant i \leqslant t$, find u_i such that
  $$\sum_{i=0}^{t} u_i Y_i = d.$$
  3. Set $\hat{a} = \frac{1}{d}\sum_{i=0}^{t} u_i Y_{i+1}$, and $\hat{b} = X_1 - \hat{a}X_0$. This stage will give $X_{i+1} = \hat{a}X_i + \hat{b} \bmod m$ for all $i \geqslant 0$.

- **Stage II:** In this stage, we begin predicting X_{i+1} and, if necessary, modifying m. When a prediction X_i is made, the actual value will be available to the inference algorithm. Initially, we set $i = 0$ and $m = \infty$ and assume X_0 and X_1 are available (we can reuse the numbers used in the previous stage). Repeat the following steps:
  1. Set $i = i + 1$ and predict
  $$X_{i+1} = \hat{a}X_i + \hat{b} \bmod m.$$
  2. If X_{i+1} is incorrect, $m = gcd(m, \hat{a}Y_{i-1} - Y_i)$.$X_i$ can be inferred *in the limit*.

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 $m \geqslant 2$ 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).

Table 1
Results of Plumstead's algorithm

$\lvert m \rvert$ 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

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.

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.

4.3. Key selection

Based on the results illustrated in Table 1, the moduli we choose is a 16-byte prime. This could also facilitate the selection of suitable X_0, a, b, and m that satisfy the security requirements, as we show later. By the Prime Number Theorem that the number of positive prime less than n is asymptotic to $n/\ln n$, the density of 16 byte primes is about $\frac{1}{\ln 2^{128}} = 0.0127$. Here, ln is the natural logarithm whose base is e. Therefore, on average we can successfully pick up a prime within about 100 random selections. Then, we randomly assign numbers less than m to X_0 without further imposing any restriction except for some trivial values such as 0 or 2^k. There is no concern about the size of the cycle in the sequence generated, since a 16-byte prime as the modulus is very likely to generate unrepeated numbers within the length of a regular data message, which is usually short in WSNs.

In our scheme, we only keep X_0 as the secret shared between two nodes. a, b, and m can be made open. They could be treated as the WSN parameters. Careful selections of a, b, and m are needed, though, in order to achieve the maximum security using the LCG. In this respect, we apply Hull and Dobell's Theorem [19] as follows.

(a) *Hull and Dobell's Theorem:* The linear congruential sequence $X_0, X_1, X_2, \ldots$ generated by

$$X_{n+1} = aX_n + b \bmod m \tag{2}$$

has a period (the number of integers before the sequence repeats) of length m if the following conditions hold:

(1) $\gcd(c, m) = 1$: The only positive integer that (exactly) divides both m and c is 1. That is, c is relatively prime to m.

(2) $p|(a-1)$, for every prime p such that $p|m$: If p is a prime number that divides m, then p divides $(a-1)$.
(3) If $4|m$, then $4|(a-1)$: If 4 divides m, then 4 divides $(a-1)$.

Since the results of Plumstead's algorithm suggest that the LCG can be broken almost in a constant number of observed random numbers, our system is not more secure if we keep all parameters $a, b, m,$ and X_0 in secret. In this respect, we make them public except X_0. Our goal is to hide all random numbers from the adversary and set up a system that chosen-plaintext attack cannot be conducted. The security of our system then does not rely on the cryptographic strength of the LCG (which is extremely weak). Instead, we rely on the LCG's statistical randomness, i.e., uniformality and period of repetition. Besides the LCG, such statistical properties of any PRNG can be easily tested. Based on Hull and Dobells Theorem, the LCG can reach such maximal statistical randomness under the conditions listed above, which are rather easy to achieve. When the period of the LCG reaches its maximum value, the chance to guess a right X_0 is $1/m$. Also, in practice, the chance that two nodes have their sequence overlapped is slim when m is sufficiently large. In our case, m has at least 128 bits.

Since X_0 is the only shared secret, key pre-distribution is relatively easier. For example, the Blom key predistribution scheme [20] can be used to allow *any* pair of nodes to compute one secret shared key (*single* key space) (It is worth noting that, based on the Blom key predistribution scheme, Du et al. [21] proposed a pairwise key predistribution scheme using *multiple* key spaces). In this paper, we focus on the discussion of a LCG-based scheme. X_0 can be any number in $\mathscr{Z}_m = 0, 1, \ldots, m-1$. If the environment is detected more hostile, our idea is still workable but a more complicate yet more cryptographically secure PRNG should be used to replace the LCG. Therefore, in this respect, the system is not more secure if we keep a, b, and m the shared secret.

In order to speed up our modulus operation and reduce the computing overhead for each sensor node, we make the following requirement for the multiplier a and the modulus m:

$$2^{63} < a < 2^{64} \quad \text{and} \quad 2^{127} < m < 2^{128}.$$

4.4. LCG based security protocols in WSNs

In this section, we briefly introduce our LCG based security protocols for WSNs.

Our proposed cipher to encrypt a 16 Byte packet is illustrated in Fig. 2(a). In Fig. 2(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.

a. Step 1 – Random Number Generation: We use the LCG to generate the random number. Given a 16 byte block cipher, one 16 byte random number, X_1, is needed.
b. Step 2 – Stage I: Suppose p_1 and p_2 are the plaintext message to be encrypted using this block cipher. Each p_i is 8 bytes. We embed the pseudo-random number X_1 into the plaintext message in the following way.For example, let *Wirelesssensor* (16 bytes) be the message to be encrypted. So $p_1 = Wireless$, and $p_2 = sensor$. The first three characters of p_1 are $W = 87$, $i = 105$, and $r = 114$. The embedding operations are simply the addition modulo 256. If

$$X_1 = 10\ 5A\ FB\ 11\ FC\ BB\ 00\ 11\ 22\ 33\ 44\ 55\ 66\ 77\ 88\ 99_h$$

The values of the first three bytes are $10_h = 16$, $5A_h = 90$, and $FB_h = 251$. Therefore, the values of the first three ciphertext characters encrypted are:

$$87 + 16 \bmod 256 = 103$$
$$105 + 90 \bmod 256 = 195$$
$$114 + 251 \bmod 256 = 109$$

As illustrated in Fig. 2(a), C_1, and C_2 are the scrambled text after X_1 is embedded. Each C_i is also 8 bytes.
c. Step 3 – Permutation: X_1 is broken into 16 1 byte random numbers, denoted as $B_0, B_1, \ldots, B_{15}$, respectively. We introduce a permutation function Π over $Z_{16} = \{0, 1, 2, \ldots, 15\}$. Let $\Pi = \pi_0\pi_1\pi_2 \ldots \pi_{15}$ be constructed as follows:
I. $\pi_0 = B_0 \bmod 16$;
II. $\pi_i = (n \bmod 16)$, for $i = 1 \ldots 15$ with n is the smallest integer such that $n \geqslant B_i$ and $\pi_i \not\subseteq \{\pi_0, \pi_1, \ldots, \pi_{i-1}\}$.
d. Step 4 – Stage II: After we obtain Π, we apply Π to C_1C_2 obtained in **Step 2** in a standard manner, i.e., the i-th byte of $\Pi(C_1C_2)$ is the π_{ith} byte of C_1C_2. Presented by 8 byte segments, let $\Pi(C_1C_2) = C\prime_1C\prime_2$, which are our final encrypted message.

Decryption is straightforward. The receiver node could generate the same X_1 that the sender generates. Using X_1, the receiver can obtain p_1 and p_2 following the backward of Fig. 2(a).

Based on an LCG based block cipher, the overall hop-by-hop security scheme is illustrated in Fig. 2(b). In Fig. 2(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.

4.5. LCG based security protocols for RFID

4.5.1. 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.

4.5.2. Length selection and security analysis

Based on this consideration, we tailor Fig. 2(b) to a more general and lightweight block cipher, as illustrated in Fig. 3. In Fig. 3, 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 $\pi_0, \pi_1, \ldots \pi_{i-1}$.

4.5.2.1. 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, $\frac{256^{2L}}{(2L)!} \approx 2^{49}$,

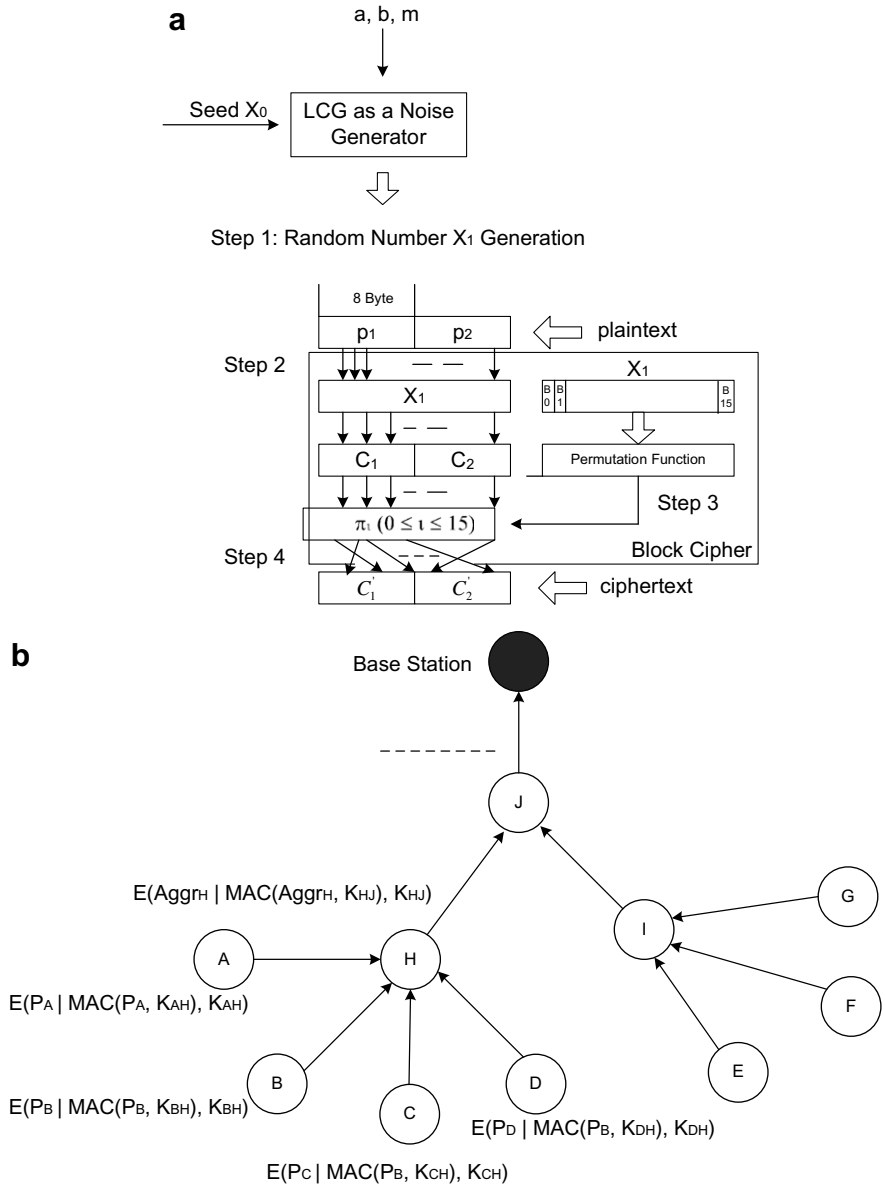

Fig. 2. LCG based hop-by-hop security protocol for WSNs. (a) Message encryption of a 16 byte packet. (b) Hop-by-hop security protocol.

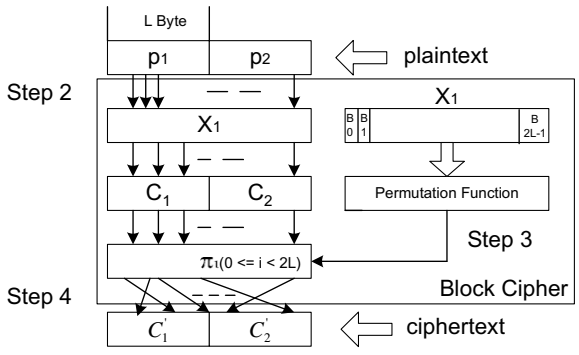

Fig. 3. LCG based block cipher for RFID.

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.

4.5.2.2. 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. 2(b) is the same as that illustrated in Fig. 3. This can facilitate security integration of WSNs and RFID systems.

4.5.3. Mutual authentication

A very basic mutual authentication protocol in RFID is presented in Fig. 4(b). In Fig. 4(b), 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 $(ID \text{XOR } LCG_{X_1}(r_1))$ 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 $(ID \text{XOR } LCG_{X_2}(X_1))$ is sent from the reader to the tag.

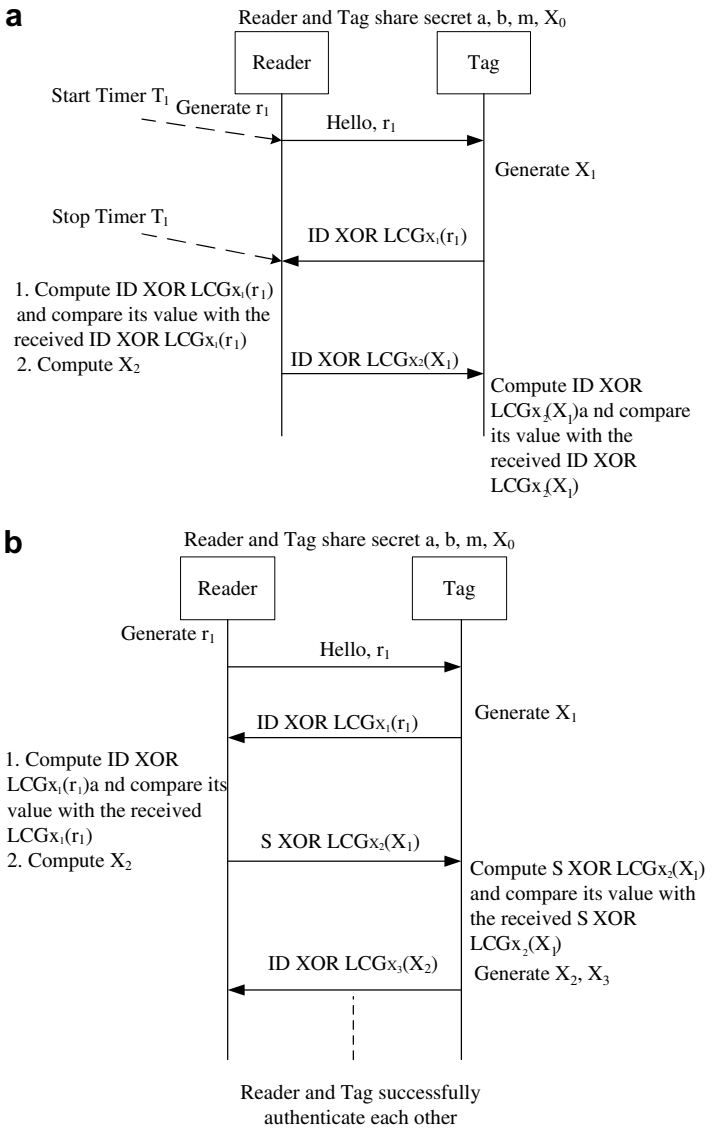

Fig. 4. LCG based hop-by-hop security protocol for RFID Systems. (a) Mutual authentication between a reader and a tag. (b) Mutual Authentication between a reader and a tag for binary tree walking scheme.

4.5.4. 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. 4. In Fig. 4, we omit the timer to make the figure better illustrated.

Similar to Fig. 4(b), 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.

4.5.5. 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 two is a left-shift operation; the operation of $(n \bmod 32)$ 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.

Table 2
Numbers of basic operations in LCG-based cipher

Operation	8-bit Operation $L=4$	8-bit Operation $L=8$	8-bit Operation $L=16$
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 3
Numbers of basic operations for generating one LCG pseudo-random number when $L=8$

LCG operations	number of LCG operation	Equivalent Operation	8-bit Operation $2L$ byte
$2L$ Byte	1	Addition	4128
Addition			
$2L$ 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 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.

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, L. Cai, Security and privacy in RFID and applications in telemedicine, IEEE Commun. Mag. (2006) 64–72.

[2] B. Sun, C.C. Li, K. Wu, 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, D. Wagner, Privacy and security in library RFID: issues, practices, and architectures, in: ACM CCS'04, Washington, DC, USA, October 2004, pp. 210–219.

[4] T. Dimitriou, A lightweight RFID protocol to protect against traceability and cloning attacks, in: IEEE SecureComm'05, Athens, Greece, September 2005, pp. 59–66.

[5] S.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, 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, D. Wagner, TinySec: a link layer security architecture for wireless sensor networks, in: ACM Sensys'04, Baltimore, MD, 2004, pp. 162–175.

[9] W. Du, J. Deng, Y.-S. Han, P. Varshney, J. Katz, A. Khalili, A pairwise key predistribution scheme for wireless sensor networks, ACM Trans. Inform. Syst. Security (TISSEC) 8 (2) (2005) 228–258.

[10] W. Du, J. Deng, Y.-S. Han, P. Varshney, A key predistribution scheme for sensor networks using deployment knowledge, IEEE Trans. Depend. Secure Comput. 3 (2) (2006) 62–77.

[11] D. Liu, Q. Dong, Detecting misused keys in wireless sensor networks, in: International Performance Computing and Communications Conference (IPCCC 2007), April 2007, pp. 272–280.

[12] S. Zhu, S. Setia, S. Jajodia, P. Ning, An interleaved hop-by-hop authentication scheme for filtering false data injection in sensor networks, in: IEEE Symposium on Security and Privacy, Oakland, CA, 2004, pp. 260–272.

[13] H. Chan, A. Perrig, D. Song, Secure hierarchical in-network aggregation in sensor networks, in: ACM CCS'06, Alexandria, VA, 2006, pp. 278–287.

[14] Y. Yang, X. Wang, S. Zhu, G. Cao, "SDAP: a secure hop-by-hop data aggregation protocol for sensor networks, in: ACM MOBIHOC'06, Florence, Italy, May 2006, pp. 356–367.

[15] D. Liu, P. Ning, W. Du, Attack-resistant location estimation in sensor networks, in: IEEE IPSN'05, Los Angeles, CA, 2005, pp. 99–106.
[16] W. Du, L. Fang, P. Ning, LAD: localization anomaly detection for wireless sensor networks, J. Parallel Distrib. Comput. (JPDC) 66 (7) (2006) 874–886.
[18] TinyOS. Available from: <http://www.tinyos.net>.
[19] D.E. Knuth, The Art of Computer Programming. Vol. 2: Seminumerical Algorithms, Addison-Wesley, 1969.
[20] R. Blom, An optimal class of symmetric key generation schemes, Advance in Cryptography EUROCRYPT, 1985 Lecture Notes in Computer Science, vol. 209, Springer-Verlag, 198, pp. 335–338.
[21] Donggang Liu, Peng Ning, Wenliang Du, Group-based key pre-distribution in wireless sensor networks, ACM Trans. Sensor Netw. (TOSN) (2008).
[22] B. Song, C.J. Mitchell, RFID authentication protocol for low-cost tags, in: ACM Conference on Wireless Network Security (WiSec'08), Alexandria, VA, 2008.