KEY MANAGEMENT ISSUES IN WIRELESS SENSOR NETWORKS: CURRENT PROPOSALS AND FUTURE DEVELOPMENTS

JOHNSON C. LEE AND VICTOR C. M. LEUNG, UNIVERSITY OF BRITISH COLUMBIA
KIRK H. WONG, JIANNONG CAO, AND HENRY C. B. CHAN,
HONG KONG POLYTECHNIC UNIVERSITY

Key management has remained a challenging issue in wireless sensor networks due to the constraints of sensor node resources. They first examine the security and operational requirements of WSNs and then review five key management protocols: Eschenauer, Du, LEAP, SHELL, and Panja.

ABSTRACT

Key management has remained a challenging issue in wireless sensor networks (WSNs) due to the constraints of sensor node resources. Various key management schemes that trade off security and operational requirements have been proposed in recent years. In this article, we first examine the security and operational requirements of WSNs and then review five key management protocols: Eschenauer, Du, LEAP, SHELL, and Panja. Eschenauer's scheme is a classical random key distribution scheme for WSNs. Du's scheme improves on Eschenauer's scheme by using key matrices. LEAP provides a highly flexible key management scheme using four types of keys. SHELL focuses on achieving high robustness, and Panja is optimized for hierarchical WSNs. LEAP, SHELL, and Panja support cluster-based operations and are more aligned with current trends as shown by the new standards, IEEE 802.15.4b and the ZigBee "enhanced" standard. Future developments likely will incorporate the features of LEAP and adjustable robustness enhancements from Eschenauer or Du; extremely security-critical applications may benefit from restructuring SHELL to ease implementation and maintenance. Developments for extremely large WSNs should consider improving Panja's scheme due to its hierarchical scalability feature.

INTRODUCTION

A wireless sensor network (WSN) is a network formed by a large number of sensor nodes, each equipped with sensor(s) to detect physical phenomena such as heat, light, motion, or sound. Using different sensors, WSNs can be implemented to support many applications including security, entertainment, automation, industrial monitoring, public utilities, and asset management. However, many WSN devices have severe resource constraints in terms of energy, computation, and memory, caused by a need to limit the cost of the large number of devices required for many applications and by deployment scenarios that prevent easy access to the devices. Such resource limitations lead to many open issues — including WSN security — which have been studied actively by researchers. Many applications require WSNs to exchange sensitive information or contain feedback processes that have high reliability requirements, and they require a high level of security to succeed. Yet, strong security is difficult to achieve with resource-limited sensor nodes, and many well-known approaches become infeasible. In this article, we explore the security issues for key management for WSNs. First, we examine the needs and requirements for key management. Then, we explore several promising key management protocols, and conclude with a discussion of future trends that may affect their development.

KEY MANAGEMENT

Before a WSN can exchange data securely, encryption keys must be established among sensor nodes. Key distribution refers to the distribution of multiple keys among the sensor nodes, which is typical in a non-trivial security scheme. Key management is a broader term for key distribution, which also includes the processes of key setup, the initial distribution of keys, and key revocation — the removal of a compromised key.

This work was supported in part by the Canadian Natural Sciences and Engineering Research Council under grant STPGP 322208-05 and by the Department of Computing, The Hong Kong Polytechnic University under account number Z09Z.

1536-1284/07/$20.00 © 2007 IEEE

Requirement	Description
Confidentiality	Nodes **should not** reveal any data to unintended recipients.
Integrity	Data **should not** be changed between transmissions due to the environment or malicious activities.
Data freshness	Old data **should not** be used as new (i.e., prevent replay attacks).
Authentication	Data used in decision-making processes **must** originate from the correct source.
Robustness	When some nodes are compromised, the entire network **should not** also become compromised. The quantitative value with which this requirement should be satisfied depends on the application.
Self-organization	Nodes **should** be independent and flexible enough to be self-organizing (autonomous) and self-healing (failure tolerant).
Availability	The network **should not** fail frequently.
Time synchronization	Collaborative node applications **need** time synchronization. Time synchronization protocols **should not** be manipulated to produce inaccurate time.
Secure localization	Nodes **should** be able to accurately and securely acquire location information.

Table 1. *Security requirements for wireless sensor networks [1].*

The need for key management starts in the link layer. An applicable link layer standard in a WSN is IEEE 802.15.4. Although this standard considers key usage for secure data transmission, it does not specify how to exchange keys securely.

The Need for Key Management

Key management, like security, is a cross-layered issue. The need for key management starts in the link layer. An applicable link layer standard in a WSN is IEEE 802.15.4. Although this standard considers key usage for secure data transmission, it does not specify how to exchange keys securely. This leaves open the key management problem that is the focus of much recent research. Besides the link layer, upper layers such as the network and application layers also must exchange keys securely. Many security-critical applications depend on key management processes to operate but also demand a high level of fault tolerance when a node is compromised. This is a challenging problem because there are many stringent requirements for key management, and the resources available to implement such processes are highly constrained.

Security and Operational Requirements for Key Management

Key management requirements can be divided into security requirements that form a subset of the overall WSN security requirements and operational requirements that act as constraints in the design and realization of key management.

Table 1 [1] shows the comprehensive security requirements of a WSN. For key management, the most critical requirements are *robustness* and *self-organization*. Although *confidentiality* and *integrity* are important, the ability to distribute secret, shared keys satisfies both requirements, and all key management schemes are able to accomplish this. Likewise, *data freshness* is typically attained by including a nonce (a cryptographic time stamp) in each packet to verify that the data is new. That approach hinges on the integrity of each data packet to ensure the nonce has not been modified, which easily can be accomplished after a secret, shared key is established. On the other hand, *self-organization* — the ability to independently self-organize and self-heal in the presence of dynamic changes in a WSN — is a requirement that is more difficult to satisfy. Normally, without the considerations of security, WSNs are designed to satisfy this property such that nodes can freely form connections around a failed node or re-establish the network automatically after it has been disturbed. However, when a key management scheme distributes certain communication keys to a few nodes, this requirement can be violated as other nodes are unable to form connections dynamically with these specific nodes for lack of proper keys.

With robustness, the problem lies with the compromise of one or more nodes. Because WSN nodes are frequently deployed in unsupervised and remote locations, physical tampering is a real threat, and the WSN must be able to withstand the compromise of some nodes. If the network uses only one key, then the compromise of one node compromises the entire network. If the network uses multiple keys, it is interesting to determine how many compromised nodes it takes to compromise the security of the entire network. We look at the distribution schemes and discuss robustness in more detail later.

On the other hand, several operational requirements exist for WSNs: *accessibility*, *flexibility*, and *scalability*. These requirements act as constraints for security design because one must ensure that they are not violated in the design and realization of a security scheme. Accessibility, the need for data to be accessible by many nodes, arises as WSNs must efficiently utilize the limited energy, computation, and memory

Requirement	Description
Accessibility	Intermediate nodes **should** be able to perform data aggregation by combining data from different nodes. Neighboring nodes **should** also be able to passively monitor event signals to prevent large amounts of redundant event signaling information.
Flexibility	Nodes **should** be replaceable when compromised. On-the-fly addition of nodes should also be supported.
Scalability	A WSN **should** concurrently support at least 3000 nodes even with the key management scheme in place.

Table 2. *Operational requirements for wireless sensor networks [1].*

resources. A popular scheme to enhance resource efficiency is data aggregation or data fusion, which combines data from other nodes with local data before routing the fused data towards the sink node. This process requires the intermediate nodes to interpret the data being sent by the other nodes. Likewise, at event detection points, neighbor nodes must be able to overhear transmissions from other nodes to ensure that only one node reports the event instead of every possible node. Later, we discuss how different schemes impact accessibility. Another crucial operational requirement is flexibility. For example, in a large WSN, nodes can frequently join or leave the network due to network expansion or battery depletion. A key management scheme must support this process with easy addition or deletion of keys. This is challenging as the lack of a central key management entity makes the addition and deletion of keys a tedious, inefficient, and more importantly, an insecure task. Lastly, the scalability of the key management scheme is an important performance factor. IEEE 802.15.4-compliant WSNs can theoretically support up to 65,536 nodes. Therefore, a key management scheme must be scalable up to this maximum network size. Although WSNs seldom exceed a few thousand nodes in practice, even this number is very challenging for some key management schemes that are less scalable. The previous discussion is summarized in Table 2.

Key Distribution Schemes

The three simplest keying models that are used to compare the different relationships between the WSN security and operational requirements are *network keying*, *pairwise keying*, and *group keying*. A detailed problem and benefit analysis is summarized in Table 3.

The network keying model has inherent advantages over the other two schemes. It is simple, easy to manage, and uses very little resources. It also allows easy collaboration of nodes since neighboring nodes can read and interpret each other's data, satisfying the self-organization and accessibility requirements very well. It is also excellent in terms of scalability and flexibility because there is only one key for the entire network, and it does not change with the addition of nodes. However, an unacceptable drawback in robustness exists. Suppose one node is compromised, and the network-wide key is exposed. With this key, an adversary can eavesdrop on all messages in the network and even inject forged messages into the network, possibly disrupting the proper operation of the network.

At the other extreme, the pairwise keying model employs $N - 1$ keys in each node, where N is the size of the network. Although this model provides the ultimate in robustness against node capture because the compromise of one node does not compromise any other node, it fails to satisfy the scalability requirement because the storage cost grows rapidly with network size. In the case of several thousand nodes, the number of keys each node must maintain becomes unmanageable. Consider the storage of $N - 1$ keys per node. The total number of distinguishable keys in the network is $N(N - 1)/2$, which grows at a rate of N^2; this is not maintainable when N is a large value. Another issue with the pairwise keying model is that it is difficult to add new nodes to the network, affecting the flexibility requirement. When a new node is added, every node must obtain a new key to communicate with it. This is a resource-intensive process that uses much more precious energy when compared with the simple preloading of a network-wide key as in the previous model. Similarly, key revocation and key refreshing suffer from the same scalability problem. Additionally, the accessibility requirement is in jeopardy as nodes cannot passively monitor event signals. Lastly, in the case of some pairwise key distribution schemes, self-organization comes into question, because they tackle the scalability problem by reducing the number of shared keys, resulting in some nodes being unable to communicate with others and compromising the self-healing and self-organizing abilities of the network.

The *group keying scheme* combines the features of both network and pairwise keying schemes. Within a group of nodes that form a cluster, communications are performed using a single, shared key similar to network keying. However, communications between groups employ a different key between each pair of groups in a manner identical to the pairwise keying scheme. Thus, for a group of nodes, the accessibility requirement is satisfied because data aggregation can occur with no additional cost while some degree of robustness is maintained. When one of the nodes is compromised, the worst-case scenario is the compromise of the entire cluster that it belongs to, which is considerably more isolated than the entire network. In terms of scalability, an acceptable trade off is possible in this scheme, because the number of keys increases with the number of groups, not with the size of the network. However, the problem with this scheme is that it is difficult to set up and also, the formation of the groups is a very application dependent process. To efficiently distribute the keys, a keying scheme would require group formation information. Furthermore, the existing IEEE 802.15.4 MAC specification has no support for group keying with its current use of access control lists [2].

Model	Description	Benefits	Problems
Network	The entire network uses one shared secret key.	• Simple • Allows data aggregation and fusion • Scalable • Able to self-organize • Flexible/accessible	• Compromise of one node compromises the entire network (lacks robustness)
Pairwise	Each specific pair of nodes shares a different key.	• Best robustness • Authenticates each node	• Nonscalable — storage, energy, computation • Unable to self-organize • Not flexible for addition/removal of nodes
Group	Each group uses a different shared key.	• Allows multicast • Allows group collaboration • Better robustness than network-wide keying • Adjustable scalability • Addition/removal of nodes possible • Able to self-organize within the cluster	• Lacks efficient storage method for group keying in IEEE 802.15.4 • Difficult to set up securely • Cluster formation information is application-dependent

Table 3. *Common key distribution schemes for WSNs.*

Promising Key Management Protocols

To realize a practical, robust keying model, researchers have proposed various key management protocols that address the problems in each of the three basic schemes discussed previously. In this section, five different key management protocols are presented and reviewed in chronological order.

Eschenauer and Gligor

Eschenauer and Gligor [3] presented one of the first key management schemes for WSNs that is elegant, simple, and provides an effective trade off between robustness and scalability. This scheme works as follows:

- Generate a large pool of keys (e.g., 10,000 keys).
- Randomly take k keys out of the pool to establish a key ring, where $k \ll N$, where N is the total number of nodes. Each node receives its own unique key ring, consisting of a subset of keys.
- When two nodes must communicate, they search for a common key within the key ring by broadcasting the identities (IDs) of the keys they have. If such a key does not exist, they attempt to communicate through a common third party, who is able to establish communications with both nodes. This phase is called path-key discovery.

As one can see, as long as the total number of keys stored in a node is less than $N - 1$, the scheme uses less storage than a pure pairwise scheme. This scheme also is scalable because the number of keys in the pool and the size of the key ring are both adjustable. Therefore, a more mission critical application can use a larger pool of keys and adjust the key ring size appropriately to be more secure. However, this scheme has some drawbacks. Compared to the newer schemes, Eschenauer's random key distribution is just a key distribution scheme. It lacks the authentication process and does not clearly define any process for revoking or refreshing keys. In addition, the dynamic handshaking process for each connection prevents any form of passive data aggregation; thus, one event detected by two neighboring nodes will result in two separate signals. There is no support for clustering or collaborative operations. If a home lighting automation application uses Eschenauer's keying scheme, turning off all the lights on one floor would entail sending a message to each light, which is rather inefficient. Lastly, since not every node is guaranteed to have a common key with all of its neighbors, there is a chance that some nodes will be unreachable [3]. Overall, Eschenauer's scheme failed to satisfy security requirement authentication and operational requirement accessibility.

Du, Deng, Han, and Varshney

In 2003, Du *et al.* proposed a key management scheme [4] based on the pairwise keying model. This model extends Eschenauer and Blom's work [5] by using the same paradigm as Eschenauer and Gligor [3] but instead of individual keys, it uses the concept of Blom's key matrix, which is an array of keys. In Du's scheme, there are k key matrices in each node, and the key matrices are distributed randomly.

Blom's model is based on the idea of a symmetric matrix multiplication, where row i column j is equivalent to row j column i. Thus, when node i calculates key ij and node j calculates key ji, the keys are identical, leading to a commonly shared secret. Blom's scheme distributes the information required for this calculation in terms of a public matrix and a private matrix.

In Du's pairwise key management scheme, instead of using only one private matrix, the sink node generates i private matrices, and each node stores a subset of these matrices in the same manner as Eschenauer's key ring. When two

The main benefit of SHELL is that it has a high robustness against node capture. Although some nodes have unique functions, the capture of that particular node does not reveal enough keys to compromise the entire network nor to disrupt the operation of the network.

nodes must communicate, they start by broadcasting the node IDs, the indices of key matrices they carry, and the seed of the column of the public matrix. If they share a common key matrix, then they can compute the pairwise secret key using Blom's scheme. If they do not share a common key matrix, they will go into a path-key discovery phase to find a common third party to route the data.

The benefit of Du's scheme is that it offers an even stronger robustness against node compromise at a reasonable scalability cost. The authors claimed that an adversary must compromise five times as many nodes compared with Eschenauer's scheme to compromise the entire network. Their analysis of scalability shows that the energy cost remains reasonable and on par with the energy cost of using the advanced encryption standard for a WSN consisting of 2^{64} nodes, which is 48 times higher than the maximum number of nodes defined in IEEE 802.15.4. The main disadvantage of this scheme is its complexity, which makes it hard to implement and increases overhead costs. Also, cluster operations are not supported because it is a pairwise keying scheme, and neither key revocation nor key refreshing are considered. The operational requirement, accessibility, is also difficult to satisfy because nodes will not be able to passively monitor communications. Lastly, compared to other, simpler schemes, Du's scheme likely uses more energy due to its computational complexity and on-demand key computation. To summarize, Du's scheme can satisfy most requirements but in the operational requirements area, it fails to satisfy accessibility and may not be competitive with simpler schemes in terms of scalability due to its higher overhead costs.

LEAP — Zhu, Setia, and Jajordia

Zhu, Setia, and Jajordia introduced the localized encryption and authentication protocol (LEAP) [6], which employs a hybrid approach. This is a *jack-of-all-trades* protocol offering network-wide, cluster/group, and pairwise keying capabilities. To accomplish this, LEAP uses four types of keys: *individual*, *group*, *cluster*, and *pairwise shared keys*. The *individual key* is unique for each sensor node to communicate with the sink node. The *group key* is a network-wide key for communication from the sink node to all sensor nodes. An authentication mechanism known as μTimed Efficient Streaming Loss-tolerant Authentication Protocol (μ*TESLA*) [7] is used for the broadcast authentication of the sink node, which ensures that packets sent with the group key are from the sink node only. The *cluster key* is used for collaborations within a cluster. An authentication mechanism known as a one-way hash-key chain that employs a non-reversible mathematical operation is used to ensure that the source of the packet can be authenticated without precluding passive data aggregation. Lastly, the *pairwise shared key* is used for secure communications between neighboring nodes.

LEAP uses a pre-distribution key to help establish the four types of keys. The individual key is first established using a function of a seed and the ID of the node. Then, in the pairwise shared key phase, a neighbor discovery process is initiated, and nodes broadcast their IDs. The receiving node uses a function, seeded with an initial key, to calculate the shared key between it and all of its neighbors. Afterwards, the initial key and any intermediate keys that were generated are erased. Thirdly, the cluster key is distributed by the cluster head using pairwise communication secured with the pairwise shared key. Lastly, for distributing the network-wide group key, the sink node broadcasts it in a multihop, cluster-by-cluster manner starting with the closest cluster.

LEAP has many advantages that satisfy the requirements of WSNs. First, it has μ*TESLA* and one-way key chain authentication, as well as key revocation and key refreshing. The accessibility requirement also can be easily satisfied by encrypting data that requires aggregation with the cluster key. The fine granularity it supports enables data to be encrypted at the correct level (i.e., key level) to ensure reasonable security is achieved without prohibiting data fusion or aggregation. The scalability of LEAP can be analyzed in terms of computational cost and storage cost. The computational cost of LEAP is inversely proportional to the number of nodes and directly proportional to the number of neighbors (i.e., node density) [6] because the higher the density of the network, the more connections are formed per cluster. The storage cost also is quite reasonable as pairwise keying is used only for one-hop neighbors. It is apparent that LEAP satisfies both the security and operational requirements very well. The only drawback with LEAP is that it assumes the sink node is never compromised.

SHELL — Younis, Ghumman, and Eltoweissy

The Scalable, Hierarchical, Efficient, Location-aware, and Light-weight (SHELL) protocol [8] is a complicated cluster-based key management scheme published recently. It is influenced by LEAP with its use of multiple types of keys but introduces a new distributed key management entity. Each cluster has its own distributed key management entity residing in a non-cluster-head node. Thus, the operational responsibility and key management responsibility are separated, leading to a better resiliency against node capture. Because there are multiple different entities and over seven types of keys, the key set up and communication processes are too complex to be described in detail in this article. Interested readers are referred to [8].

The main benefit of SHELL is that it has a high robustness against node capture. Although some nodes have unique functions, the capture of that particular node does not reveal enough keys to compromise the entire network nor to disrupt the operation of the network. For instance, the capture of a key-generating gateway node, a key management entity, does not compromise the network because it does not contain the key between the cluster head and the cluster nodes. Likewise, due to the distributed feature, there are at least two key-generating gateway nodes and the disruption of one does not hinder the operation of the network. In addition, SHELL accounts for the processes for node addition, replacement, and refreshment of

Scheme	Date	Structure	Key generation	Description
Eschenauer	2002	Partial pairwise, random key distribution	Static	Randomly selects *k* keys out of a large pool to form a key ring. Common keys in a pair of node's key ring allow communication.
Du	2003	Pairwise, matrix	Dynamic	Selects *T* key matrices and uses matrix multiplication to compute pairwise shared key dynamically.
LEAP	2003	Hybrid: network, group, neighbor pairwise	Mostly static	Uses a predistributed key to establish four types of keys.
SHELL	2006	Group, distributed key management roles	Dynamic	Uses a distributed key management entity to generate and manage keys.
Panja	2006	Group, hierarchical network	Dynamic	Uses multiple partial keys to compute group keys dynamically.

Table 4. *Summary of key management schemes.*

keys. It also supports cluster (group) communications and does not preclude data fusion or aggregation within the clusters. However, SHELL has some drawbacks. Its structure and operation are highly complex, involving heterogeneous node operations and multiple (at least seven) types of keys. The specific network entities include the gateway node, key-generating gateway node, inter-gateway node, command node, and sensor node. The energy consumption and cryptographic overheads, although scalable, have not been compared with other schemes. Due to its increased complexity, the energy usage and cryptographic overhead are likely higher than other schemes. Finally, the implementation of such a complex protocol also may be difficult with the current programming limitations of a WSN. To conclude, SHELL focuses on satisfying the robustness and security requirements while trading off the *availability* requirement since higher complexity leads to higher energy usage and lowers the mean time between failures. In this case, failures would likely occur due to the depletion of battery energy of individual nodes.

Panja, Madria, and Bhargava

Panja *et al.* [9] recently introduced a hierarchical group keying scheme using the Tree-based Group Diffie-Hellman (TGDH) protocol. The main feature of this scheme is that each key is made up of many partial keys. By breaking up the keys into smaller components, it makes rekeying an efficient and simple task by revoking, changing, or adding one or more partial key(s).

The TGDH keying scheme works on a hierarchical WSN that has one level of general sensor nodes and multiple levels of cluster heads; that is, there can be a *head of clusters* responsible for multiple cluster heads below it in a tree-like manner. The data collection process starts with a group of sensor nodes collecting data from a region of interest and sending it to the nearest cluster head. The cluster head then aggregates the data to reduce the size and overhead and sends it to its parent. The parent, if it has multiple children, repeats the process of data aggregation and forwards the data to its parent until the sink node is reached.

To establish the keys in this hierarchical tree-based WSN, two separate schemes are used: intra-cluster and inter-cluster keying. The intra-cluster keying process starts with the leaf nodes sending each of their partial keys to the parent. Then the parent calculates its own partial key and combines the partial keys together to form the cluster key. The parent node then broadcasts this cluster key to its leaf nodes. All communications are encrypted with a pre-distributed key to provide confidentiality during this early stage. Afterwards, the inter-cluster keying is initiated. This process is very similar to that of intra-cluster keying except that intermediate keys of the parents (children of the next higher level) are used instead of the partial keys.

The advantage of this scheme is that compared to SHELL, it is simple and elegant and hence, easy to implement. Panja *et al.* also simulated the performance of their scheme in comparison with Security Protocols for Sensor Networks (SPINS) [7], a keying protocol that establishes secure one-to-one communication in a WSN. The results are promising with fast and scalable key delivery time and energy usage. Also, by using small partial keys, the storage and computational costs are reduced. This is especially significant for the leaf nodes that frequently have the least amount of resources at their disposal. The drawback of Panja's scheme is that although key revocation and key refreshing processes are addressed, node addition and replacement are not considered explicitly. In addition, its security robustness against the compromise of the initial pre-distributed key has not been analyzed. Further, the security strength of the low complexity keys at the leaf nodes, for example, against brute-force attacks, has not been proven. To conclude, Panja's scheme trades off robustness to better satisfy the self-organization, accessibility, flexibility, and scalability requirements. The use of a tree-based hierarchical structure also ensures that this protocol is very scalable.

Summary

The promising key management protocols surveyed in this article are summarized in Table 4. Table 5 and Table 6, respectively, compare the performance, advantages, and disadvantages of these protocols.

Scheme	Simplicity	Scalability	Robustness	Storage efficiency[1]
Eschenauer	H	M	L	M/H
Du	L	L	H	L/M
LEAP	M	M	M	M
SHELL	L	M	H	L/M
Panja	M	H	M	M/H

Legends: H = High, M = Mid, L = Low. For each variable, the higher the value, the better performance it has.

[1] Storage cost may change with different factors. We can only estimate the relative cost based on available information.

Table 5. *Relative performance of key management schemes [10].*

Trends in Key Management

Although there are many papers published on various keying schemes, whether or not they will be implemented or used in practice depends on market demands. A proposed scheme likely will be adopted in practice if it targets open standards. Recently, both the IEEE 802.15 task group and the ZigBee Alliance have released newly revised standards. Despite the fact that the new IEEE 802.15.4b still does not specify a key management scheme, it clarifies ambiguities and enhances many features compared with the original standard. In this section, we discuss the security related changes in 802.15.4b and the features relevant to key management in the newly enhanced ZigBee standard 2006.

The 802.15 task group 4b was chartered to provide specific enhancements and clarifications to the original 802.15.4-2003 standard. The revised standard was published in September 2006 as 802.15.4-2006. In terms of the security changes, the standard introduces a new counter with cipher block chaining mode* (CCM*) cipher suite mode, which incorporates the *confidentiality-only* and *authentication-only* modes that were provided, non-securely, by cipher block chaining (CBC) and counTeR (CTR) cipher suite modes in the original standard. Another new feature of the 4b standard that is relevant to key management is secure group keying. Many applications can benefit from secure broadcasting or multicasting abilities. Unfortunately, the secure broadcasting mechanism in 802.15.4-2003 is actually insecure because it does not provide data freshness and is vulnerable to replay attacks. The replay attack vulnerability can be exploited against applications that require secure broadcast, such as home automation. Imagine the devastating effect of a replayed *lights-off* broadcast right before a home invasion. Hence, replay resistant broadcasting has been introduced in 4b using frame counters on a per-device basis, and the feature is made more flexible with the introduction of group keying. Many other small clarifications were proposed, but only the most relevant ones were presented here.

The ZigBee Enhanced standard was released in September 2006 and offers improvements and new features. The most relevant feature is the support of group devices and targeted broadcasts. Like 802.15.4b, the ZigBee Alliance also has treated group device support with a high priority. In addition, the new targeted broadcast feature that can reach a specific subset of devices (routers, end nodes, sleeping nodes, currently awake nodes) is important to key management scheme selection because it adds a new functional requirement. It is worthwhile to take these new features into account when considering the various key management schemes.

Based on the new trends in 802.15.4b and the ZigBee Enhanced standard, one can easily assume that a purely random or pairwise keying scheme like Eschenauer [3] and Du [4] would not be commercially viable. The keying schemes that offer group or multicast abilities are much more compatible with industry trends. Thus, the development of a key management scheme that incorporates the flexible network, pair, and cluster abilities found in LEAP, along with the adjustable robustness of Eschenauer or Du may be desirable. The main issue with SHELL is its high complexity; further developments can target streamlining of its processes and reduction of overheads. Panja's scheme also has some advantages for extremely large networks, and one might want to enhance Panja's scheme with the multiple-level keying of LEAP, for added flexibility, or with Eschenauer's or Du's scheme, for increased robustness. In any case, the trade offs presented earlier remain true, and the two new industry standards provide a good guideline to the acceptable essential services that a viable key management solution must provide.

Conclusion

In this article, we reviewed five key management schemes starting with the classic Eschenauer scheme and moving to the more recent schemes published in 2006. It is clear that numerous trade offs exist between different key management schemes, and the vast number of proposals make it difficult to compare them. Recent trends also show that cluster or group operation is a required feature that has been considered by many recent key management proposals including LEAP, SHELL, and Panja's scheme. Each one of these schemes has its own strengths, such as the LEAP adjustable security level, the strong robustness of SHELL, and Panja's hierarchical scalability. One must consider the trade offs carefully when selecting key management schemes. For instance, although SHELL offers a high robustness, it is also far more complex than the other two and thus, may be difficult to implement. LEAP, on the other hand, has the flexibility to switch between group, network, and pairwise keying on a per packet basis, but the protocol must be studied further for security weaknesses. Future developments could incorporate the flexibility of LEAP with the adjustable robustness offered by Eschenauer or Du's scheme. For security-critical applications, SHELL seems to offer the highest robustness, but it may be further improved to reduce imple-

Scheme	Advantage	Disadvantage
Eschenauer	• Uses less storage than a pure pairwise scheme • Adjustable robustness trade-off with storage cost • Simple and implementable	• No authentication available • No support for cluster operations • Some nodes may not be reachable • Low accessibility
Du	• Provides node authentication in an Eschenauer-like scheme • Energy cost remains reasonable • Excellent robustness	• High complexity • No support for cluster operations • High relative energy cost • Low accessibility
LEAP	• Supports cluster, pairwise, and network-wide operations (good for data fusion/aggregation) • Can detect and isolate compromised nodes quickly using μTELSA and one-way key chain hashing authentication • Reasonable complexity and scalability	• Security during the startup key establishment process may be weak • Storage cost is high for a small number of nodes due to the use of four types of keys • Assumes the sink node is never compromised
SHELL	• Supports addition and replacement of nodes • Refreshes keys using only a few messages • Utilizes distributed key management entity • High robustness • High accessibility	• Complex operations with highly heterogeneous node operations • Storage cost is high for a low number of nodes due to the use of seven or more types of keys • Higher cryptographic overhead • Overall energy cost is higher due to complex operations
Panja	• Simple and elegant • Fast and easy key refreshing process • Highly scalable using TGDH scheme • Low storage cost for leaf nodes • Ideal for shorter-term sensor networks	• Nonadjustable security strength • Does not clearly address key revocation or node addition • Long-term eavesdropping may present a threat if initial key is compromised • Leaf node security using small partial keys may not be strong enough against analysis by widely available high computational platforms

Table 6. *Comparison of advantages and disadvantages of key management schemes.*

mentation complexity. For extremely large WSNs, improving Panja's scheme to take advantage of its highly scalable hierarchical feature may prove attractive.

References

[1] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and Approaches for Distributed Sensor Security," NAI Labs tech. rep. 00-010, 2000.

[2] N. Sastry and D. Wagner, "Security Considerations for IEEE 802.15.4 Networks," *Proc. 2004 ACM Wksp. Wireless Sec.*, 2004, pp. 32–42.

[3] L. Eschenauer and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. 9th ACM Conf. Comp. and Commun. Sec.*, 2002, pp. 41–47.

[4] W. Du *et al.*, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *Proc. 10th ACM Conf. Comp. Commun. Sec.*, 2003, pp. 42–51.

[5] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Proc. EUROCRYPT '84 Wksp. Advances in Cryptology: Theory and App. of Cryptographic Techniques*, 1985, pp. 335–38.

[6] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *Proc. 10th ACM Conf. Comp. and Commun. Sec.*, 2003, pp. 62–72.

[7] A. Perrig *et al.*, "SPINS: Security Protocols for Sensor Networks," *Wireless Network*, vol. 8, 2002, pp. 521–34.

[8] M. F. Younis, K. Ghumman, and M. Eltoweissy, "Location-Aware Combinatorial Key Management Scheme for Clustered Sensor Networks," *IEEE Trans. Parallel and Distrib. Sys.*, vol. 17, 2006, pp. 865–82.

[9] B. Panja, S. K. Madria, and B. Bhargava, "Energy and Communication Efficient Group Key Management Protocol for Hierarchical Sensor Networks," *SUTC '06: Proc. IEEE Int'l. Conf. Sensor Networks, Ubiquitous, and Trustworthy Comp.*, 2006, pp. 384–93.

[10] S. A. Camtepe and B. Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: A Survey," Tech. rep. TR-05-07, Dept. of Comp. Sci., Rensselaer Polytechnic Inst., 2005.

Biographies

JOHNSON C. LEE (johnsonl@ece.ubc.ca) received his B.A.Sc. (with distinction) from the University of British Columbia, Canada, in 2005 and is currently pursuing his M.Eng. degree at the same university. During his undergraduate studies, he received the NSERC USRA Scholarship to work on communication security for SSL with Dr. Konstantin Beznosov. His research interests are communication security, sensor networks, distributed computing, and software project management.

KIRK H. M. WONG (cshmwong@comp.polyu.edu.hk) received his B.Sc. (Honors) from York University, Canada, in 1986, and his M.Sc. from Hong Kong University of Science and Technology in 1999. Currently, he is a Ph.D. student in the Department of Computing at Hong Kong Polytechnic University. He also works at the Department of Information and Communications Technology at the Hong Kong Institute of Vocational Education as a lecturer. His research interests include security and positioning in wireless sensor networks.

HENRY C. B. CHAN [M] (cshchan@comp.polyu.edu.hk) received his B.A. and M.A. degrees from the University of Cambridge, England, and his Ph.D. degree from the University of British Columbia, Canada. Currently, he is an associate professor in the Department of Computing at Hong Kong Polytechnic University. His research interests include networking/communications, wireless networks, e-commerce, Internet technologies, and mobile computing. From October 1988 to October 1993 he worked with Hong Kong Telecommunications Limited, primarily on the development of networking services in Hong Kong. Between October 1997 and August 1998 he worked with BC TEL Advanced Communications on the development of high-speed networking technologies and ATM-based services. He has authored/co-authored a textbook on e-commerce published by Wiley, a book chapter in the *Internet Encyclopedia* (Wiley), and more than 50 journal and conference papers. He is currently serving as an executive committee member of the IEEE Hong Kong Section Computer Chapter.

VICTOR C. M. LEUNG [F] (vleung@ece.ubc.ca) received a B.A.Sc. from the University of British Columbia, Canada, in 1977 and was awarded the APEGBC Gold Medal as the head of the graduating class in the Faculty of Applied Science. He attended graduate school at the same university on an NSERC Postgraduate Scholarship and completed a Ph.D. in electrical engineering in 1981. Currently, he holds the positions of professor and Telus Mobility Research Chair in the Department of Electrical and Computer Engineering at the University of British Columbia. His research interests are in mobile and wireless networks. From 1981 to 1987 he was a senior member of technical staff at MPR Teltech Ltd., Canada. He was a lecturer in electronics at the Chinese University of Hong Kong in 1988. He is an editor of *IEEE Transactions on Wireless Communications*, an associate editor of *IEEE Transactions on Computers*, and an associate editor of *IEEE Transactions on Vehicular Technology*. He has served on the organizing and technical program committees of numerous conferences. He is a registered Professional Engineer in British Columbia, Canada.

JIANNONG CAO [M'93-SM'05] received a B.Sc. degree in computer science from Nanjing University, China, in 1982, and M.Sc. and Ph.D. degrees in computer science from Washington State University in 1986 and 1990, respectively. He is currently a professor in the Department of Computing at Hong Kong Polytechnic University, Hung Hom. He is also director of the Internet and Mobile Computing Laboratory in the department. His research interests include parallel and distributed computing, networking, mobile and wireless computing, fault tolerance, and distributed software architecture. His recent research has focused on mobile and pervasive computing systems, developing testbeds, protocols, middleware, and applications. Before joining Hong Kong Polytechnic University, he was on the faculty of computer science at James Cook University, the University of Adelaide in Australia, and City University of Hong Kong. He has published over 200 technical papers in the previously mentioned areas. He is a senior member of the China Computer Federation and a member of ACM. He is also a member of the IEEE Technical Committees on Distributed Processing, Parallel Processing, and Fault Tolerant Computing. He has served as a member of editorial boards of several international journals, a reviewer for international journals and conference proceedings, and an organizing/program committee member for many international conferences.