A Key Management Scheme for Large Scale Distributed Sensor Networks

Yong Ho Kim*, Hwaseong Lee, Dong Hoon Lee, and Jongin Lim

Center for Information Security Technologies (CIST),
Korea University, Seoul, Korea
{optim, hwaseong, donghlee, jilim}@korea.ac.kr

Abstract. To guarantee secure communication in wireless sensor networks, secret keys should be securely established between sensor nodes. Recently, an efficient security mechanism was proposed for large-scale distributed sensor networks by Zhu, Setia, and Jajodia. In their scheme, each node uses a single initial key to establish pair-wise keys and erases the key after key setup. If the key is compromised during key setup, however, the entire network will be compromised. Therefore, the performance overhead during key setup is very important for the speedy key establishment. In this paper, we propose a modified scheme which reduces the performance overhead during key setup and has provable security after key setup.

Keywords: security, key management, wireless sensor networks

1 Introduction

Wireless sensor networks are well recognized as a new paradigm for future communication. Sensor networks consist of a huge number of battery powered and low-cost devices, called sensor nodes. Each sensor node is equipped with sensing, data processing, and communicating components [1, 3].

To provide secure communication within wireless sensor networks, it is essential that secret keys should be securely established between sensor nodes. The shared secret key may later be used to achieve some cryptographic goals such as confidentiality or data integrity. However, due to limited resources of sensor nodes, the traditional schemes such as public key cryptography are impractical in sensor networks. Furthermore, the position of sensor nodes (hence, the neighbors of nodes) cannot be pre-determined since they are randomly deployed in unattended areas. Due to this restriction, most schemes are based on the pre-distribution of potential keys.

Another security concern in sensor networks is resilience against node capture. Sensor nodes are deployed randomly in hostile areas where they are exposed to the risk of physical attacks. For instance, an attacker can capture sensor nodes to obtain secret information stored within memory of the nodes. The ultimate

* This work was supported by the Brain Korea 21 Project in 2006

goal may be to acquire *perfect resilience* [4] which means that even if a node is captured, it provides no information about links that it is not directly involved in.

Related Works. When we consider the design of key distribution schemes, the simplest method is to embed a single network-wide key in the memory of all nodes before nodes are deployed. In this case, however, the entire network can be compromised if a single node is captured. Another extreme method is that each node in a network of n nodes shares a unique pair-wise key with every other node in the network before deployment. This requires memory for $n - 1$ keys for each sensor node. Therefore, these two methods are unsuitable for wireless sensor networks.

Perrig et al. presented SPINS [10], security protocols for sensor networks. In SPINS, each node shares a secret key with a base station and establishes its pair-wise keys through the base station. This architecture satisfies the small memory requirement and offers perfect resilience against node capture. However, because the base station should participate in every pair-wise key establishment, SPINS requires significant communication overhead and does not support large-scale networks.

Zhu et al. presented LEAP [11], an efficient key management method. All the nodes in LEAP save an initial key before deployment. After deployment, each node establishes a pair-wise key from the initial key and then erases the initial key securely. Although LEAP is very efficient, it must assume that it is difficult for the initial key to be exposed by node capture during initial key setup.

Eschenauer and Gligor presented a random key pre-distribution scheme for pair-wise key establishment [8] in which a key pool is randomly selected from the key space and a key ring, a randomly selected subset from the key pool, is stored in each node before deployment. A common key in two key rings of a pair of neighbor nodes is used as their pair-wise key. Their scheme has been subsequently improved by Chan et al. [4], Liu and Ning [9], and Du et al. [6, 7]. However, these schemes require a significant pre-computation phase as well as a large amount of memory.

Contributions. The main contributions of our approach can be summarized as follows:

– Reduced time and cost to establish pair-wise keys. The security of LEAP depends on the success of attacks during key setup. For this reason, the performance overhead of key setup is very important for security as well as efficiency. Our scheme reduces the possibility of being attacked and consumes less energy by cutting down the time and saving the energy during key establishment.

– Provable security. We prove the security of our scheme after key setup. The proof is based on one outlined by Bellare et al. [2]. Assuming that a message authentication code(MAC) is secure against forgery under a chosen-message attack, our scheme has provable security.

– Scalability. Our scheme is suitable to a large network. We note that other schemes [4, 10] with perfect resilience against node capture are not scalable. To be scalable, SPINS [10] requires significant communication overhead because of the participation of a base station in key setup, and the scheme by Chan et al. [4] requires a large amount of memory for each node.

Organization. The rest of the paper is organized as follows. Section 2 shows the notation used in this paper. We give an overview of LEAP in Section 3. We propose our scheme and analyze its performance and security in Section 4. Finally, we conclude our paper in Section 5.

2 Notation

We list the notations used in the paper below:

Notation	**Description**
n	the expected number of neighbor nodes within communication radius of a given node
$\|\|$	concatenation operator
K_I	the initial key
K_u	the master key of u
K_{uv}	the pair-wise key between u and v
$E(K, \cdot)$	symmetric encryption function using key K
$F(K, \cdot)$	pseudo-random function using key K
$\mathsf{MAC}(K, \cdot)$	message authentication code using key K

3 LEAP [11]

Unlike previous schemes, LEAP supports four types of keys for each node. The four types of keys are as follows: an individual key, a pair-wise key, a cluster key, and a group key. First, an individual key is a shared key between each node and the base station. Second, a pair-wise key is shared between a node and its neighbor node which is only one-hop away. Third, a cluster key is a key between a node and its all neighbor nodes which are also only one-hop away. Last, a group key is one key shared by all nodes in the network.

We now give a detail of each establishment of four keys. An individual key is pre-loaded into each node before deployment; the security of this key is not considered since we assume the base station to be secure. A group key is also pre-loaded and then updated using cluster keys. A cluster key is established and updated, using pair-wise keys. Therefore, it is the security of pair-wise keys that guarantees the security of LEAP.

Now, we will describe the four phases to establish pair-wise keys for sensor nodes.

Key Pre-distribution. Before nodes are deployed, the same initial key K_I is stored on each node. Each node can derive a master key $K_u = F(K_I, u)$, using the initial key with a pseudo-random function.

Neighbor Discovery. After deployment, each node broadcasts a message consisting of its ID and a nonce it selects randomly. In return, neighbor nodes retransmit their ID and MAC that is constructed using their master key. The source is authenticated by verifying the MAC.

$$\begin{aligned} u \longrightarrow * &: \quad u, Nonce_u \\ v \longrightarrow u &: \quad v, \mathsf{MAC}(K_v, Nonce_u \| v) \end{aligned}$$

Pair-wise Key Establishment. After source authentication, the pair-wise key in each node is established through information from the neighbor nodes and a pseudo-random function. Nodes u and v use their pair-wise key as $K_{uv} = F(K_v, u)$.

Key Erasure. After key setup, the initial key K_I and all the master keys of neighbor nodes are completely erased.

When a pair-wise key is established in LEAP, there is an assumption that $T_{min} > T_{est}$. The T_{min} is the minimum time that it takes an attacker to obtain secret information from a sensor node. The T_{est} is the time required for the deployed nodes to actually detect their neighbor nodes. This assumption shows that pair-wise keys are established before an attacker captures some nodes and obtains critical information from them. It is needed to check whether this assumption is practicable or not. The transmission rate is 19.2kbps [5] and the transmitted message is very short (a total of 12 bytes when the node ID and its MAC are 4 bytes and 8 bytes respectively). Hence, the assumption is persuasive in the case where the nodes are initially deployed in the network.

4 Our Scheme

In LEAP, the entire network will suffer a severe loss if an initial key is exposed to an attacker during key setup. Hence, early key establishment must be completed quickly in order to strengthen the security of LEAP.

4.1 Initial Key Setup

In LEAP, each node executes communication at $O(n)$ until the initial key is deleted, whereas in our scheme, the communication required in each node is a single broadcast in the neighbor discovery phase. In general, more time and more energy are required in communicating than computing the symmetric functions. In an example of SPINS [10], the communication cost is about 97% while computation cost is just less than 3%. Therefore, it is much more efficient to reduce the communication cost than to improve the computation cost. Therefore our experiment is very meaningful. The four phases to establish a pair-wise key are as follows:

Key Pre-distribution. In our scheme, like LEAP, a single initial key K_I is pre-loaded in all the nodes before deployment. A sensor node u computes its master key $K_u = \mathsf{MAC}(K_I, u)$ that will be used later to establish pair-wise keys with new nodes.

Neighbor Discovery. The sensor node u selects a nonce $Nonce_u$ randomly and computes a MAC value such as $\mathsf{MAC}(K_I, Nonce_u||u)$. Then, the sensor node broadcasts a message consisting of its node ID, $Nonce_u$, and $\mathsf{MAC}(K_I, Nonce_u||u)$. After this phase, all the nodes in the network can obtain the IDs of their neighbor nodes. As each node verifies the MACs of its neighbor nodes, it can authenticate the initial key and their IDs. In LEAP, the master key of each node is used as a MAC key but it is sufficient to use the initial key as the MAC key because an attacker who cannot derive a master key is also unable to know the initial key.

Pair-wise Key Establishment. Collecting the IDs of its neighbor nodes, node u can compute $K_{uv} = \mathsf{MAC}(K_I, u||v)$ to use as a pair-wise key with its neighbor node v (if $u < v$). The pair-wise key is directly derived from the initial key without computing the master key of its neighbor nodes.

Key Erasure. In this phase, each node erases the initial key.

This scheme improves the security and efficiency of LEAP. In the neighbor discovery phase, our scheme requires a single broadcast so that the amount of communication is reduced considerably. Also, in LEAP, each node has to derive the master key of neighbor nodes to verify their MACs while this is not necessary in our scheme in that we generate the MAC with an initial key. Hence, in our scheme, both communication and computation costs are reduced.

Now that the broadcast message includes MAC values, the neighbor nodes can verify whether the party is sound through each other MAC values. In case that key confirmation is needed between the neighbor nodes, any future messages encrypted and authenticated with the pair-wise key can implicitly achieve the same effect.

4.2 Node Addition after Initial Key Setup

For a very large sensor network, node addition must be feasible anytime. In LEAP, it is difficult for a new node to establish a pair-wise key with old nodes because old nodes erased the initial key after key setup. However it is possible to establish a pair-wise key with old nodes using the master key derived from the initial key and their ID.

$$\begin{aligned} w \longrightarrow u : &\quad w, Nonce_w \\ u \longrightarrow w : &\quad u, \mathsf{MAC}(K_u, Nonce_w || u) \end{aligned}$$

A new node w is deployed with the initial key K_I pre-loaded. The new node w detects its old neighbor node u and can establish the pair-wise key $K_{wu} = F(K_u, w)$ from master key $K_u = F(K_I, u)$ of u. This method does apply to our scheme except the pair-wise key $K_{wu} = \mathsf{MAC}(K_u, w)$ between the new node w and the old node u.

The new node can quickly establish the pair-wise key with working nodes but not with sleeping nodes without some delay. The reason is that sleeping nodes cannot respond to the request of the new node until their state changes from sleeping mode to working one. If working nodes inform a new node of their neighbor nodes, the new node can establish pair-wise keys in advance through collecting IDs of sleeping nodes before sleeping nodes change their mode to working one. In that case, it is needless for new node to save the initial key until sleeping nodes convert to working mode. Therefore, the new node can establish pair-wise keys with its neighbor old nodes. In our scheme, the addition method of LEAP can be used since old nodes have the master key derived from the initial key and their ID.

4.3 Performance Analysis

Zhu et al presented a technical report about LEAP [12]. In this report, they implemented LEAP with the following algorithms on the TinyOS platform. First, the linear-feedback shift register(LFSR) was employed to generate the pseudo-random numbers. The RC5 block cipher was used for encryption along with CBC-MAC.

Also, MAC replaced both the pseudo-random functions and the one-way functions in order to lessen the space of code in the ROM. As a result, RC5 was used for all the operations - the encryption function, CBC-MAC, the pseudo-random function, and the one-way function. Our scheme also employs a MAC function in place of a pseudo-random function for the same reason.

Comparing the computation cost during initial key setup, LEAP needs the $2n$ operations for the MAC function and another $2n$ operations for the F function. Since the F function in LEAP was replaced with the MAC function, the total number of operations for the MAC function would eventually be $4n$, while the total number of operations for the MAC function in our scheme is just $2n + 1$. Conclusively, in our scheme, computation overhead is two times more efficient than in LEAP and the efficiency of communication overhead is also about $0.75n$ times better. The table below compares the two schemes when the node ID and the nonce are each 4 bytes and the MAC is 8 bytes.

Table 1. Comparison of Communication Overhead.

	Broadcast Communication	**Unicast Communication**
LEAP	$1 \times$ (8 *byte*)	$n \times$ (12 *byte*)
Our Scheme	$1 \times$ (16 *byte*)	0

4.4 Security Analysis

In both schemes, LEAP and our scheme, the entire network will sustain a serious loss under the situation of exposing the initial key. However, our scheme has less probability of exposing the initial key because the performance overhead in our scheme, including the computation and communication overhead, is more improved than in LEAP during the key setup.

In our scheme, after key setup, the information in captured sensor nodes cannot be used to find any information about shared keys between non-captured sensor nodes. We assume that an attacker captures nodes in which initial key has been erased after key setup. Hence, she cannot acquire K_I, but can obtain MAC values which were generated using K_I. In this scenario, she attempts to acquire master keys or pair-wise keys of non-captured nodes by computing them either directly or indirectly. First, if she computes K_I from the master key $\mathsf{MAC}(K_I, u')$ or the pair-wise key $\mathsf{MAC}(K_I, u'||v')$ of a captured node u', she could acquire master key $\mathsf{MAC}(K_I, u)$ or pair-wise key $\mathsf{MAC}(K_I, u||v)$ of a non-captured node u, where v' is a neighbor of u' and v is a neighbor of u. However, this is impossible due to the one-way property of MAC functions. Second, if there are some methods for indirectly computing only master key $\mathsf{MAC}(K_I, u)$ or pair-wise key $\mathsf{MAC}(K_I, u||v)$ without using K_I, our scheme will be insecure.

To formally prove the security of our scheme, we first review the security of message authentication codes defined in [2]. A message authentication code takes as inputs a key K and a message M, and outputs a string σ.

$$\mathsf{MAC} : Key(\mathsf{MAC}) \times Dom(\mathsf{MAC}) \longrightarrow \{0,1\}^k$$

The key K is shared between a sender and a receiver. When the sender wants to send a message M it computes $\sigma = \mathsf{MAC}(K, M)$ and transmits the pair (M, σ) to the receiver. The receiver re-computes $\mathsf{MAC}(K, M)$ and verifies that this equals the value σ. An attacker is allowed to mount a *chosen message attack*(cma) in which it can obtain MACs of messages of its choice. If it outputs a valid pair (M, σ) which was not a query to its MAC oracle, then it will be considered successful.

Definition 1. Consider the following experiment.

Experiment $\mathsf{Forge}^{\mathsf{cma}}_{\mathsf{MAC}}(A)$

$K \overset{R}{\leftarrow} Key(\mathsf{MAC})(k)$

```
    $(M, \sigma) \leftarrow A^{\mathsf{MAC(K,\cdot)}}$
    If $\mathsf{MAC}(K, M) = \sigma$ and $M$ was not a query of $A$ to its oracle
      then return 1 else return 0
```

Now let $\mathsf{Succ}^{\mathsf{cma}}_{\mathsf{MAC}}(A) \overset{def}{=} Pr[\mathsf{Forge}^{\mathsf{cma}}_{\mathsf{MAC}}(A) = 1]$. Then an advantage function of MAC is defined as follows:

$$\mathsf{Adv}^{\mathsf{cma}}_{\mathsf{MAC}}(q, t) \overset{def}{=} \underset{A}{max}\{\mathsf{Succ}^{\mathsf{cma}}_{\mathsf{MAC}}(A)\}$$

where the maximum is taken over all A with execution time t and at most q queries to the oracle $\mathsf{MAC}(K, \cdot)$. A message authentication code will be secure against *chosen message attack* if the advantage is negligible in the security parameter k.

We define the security of key management schemes against node capture in sensor networks. An attacker is allowed to mount a *chosen node capture attack*(cna) in which it can obtain master keys and pair-wise keys of captured nodes. If it outputs a key of a master key or a pair-wise key of an non-captured node, then it will be considered successful.

Definition 2. We model wireless sensor networks as a directed graph $G = (N, E)$ where $N = \{u_1, u_2, \cdots, u_{|N|}\}$ and $E = \{\langle u, v\rangle : u, v$ are neighbors and $u < v\}$. Let n be the expected degree of a node in G.

Definition 3. A sensor key management scheme (M) is secure against *chosen node capture attack* if the following advantage is negligible in the security parameter k. An attacker is allowed to use oracle $\mathsf{M}(\mathsf{G}, \mathsf{K}, \cdot)$ which for an input query $u' \in N$, responds $str \in \{0,1\}^{k(1+n)}$ which contains $(u', \mathsf{MAC}(K, u'))$ and (u', v', σ') where $\sigma' = \mathsf{MAC}(K, u'||v')$ if $\langle u', v'\rangle \in E$ or $\sigma' = \mathsf{MAC}(K, v'||u')$ if $\langle v', u'\rangle \in E$.

```
Experiment $\mathsf{Compro}^{\mathsf{cna}}_{\mathsf{M}}(B)$
    $K \overset{R}{\leftarrow} Key(\mathsf{M})(k)$
    Generate $G$ as defined in Definition 2.
    $G$ is given to $B$
    $(u, \sigma) \vee (u, v, \sigma) \leftarrow B^{\mathsf{M(G,K,\cdot)}}$
    If $(((u, \sigma) \wedge \mathsf{MAC}(K, u) = \sigma) \vee ((u, v, \sigma)$
      $\wedge((\mathsf{MAC}(K, u||v) = \sigma \wedge u < v) \vee (\mathsf{MAC}(K, v||u) = \sigma \wedge v < u))))$
      $\wedge(u, v$ were not a query of $B$ to its oracle)
    then return 1 else return 0
```

Now let $\mathsf{Succ}^{\mathsf{cna}}_{\mathsf{M}}(B) \overset{def}{=} Pr[\mathsf{Compro}^{\mathsf{cna}}_{\mathsf{M}}(B) = 1]$. Then an advantage function of M is defined as follows:

$$\mathsf{Adv}^{\mathsf{cna}}_{\mathsf{M}}(q', t') \overset{def}{=} \underset{B}{max}\{\mathsf{Succ}^{\mathsf{cna}}_{\mathsf{M}}(B)\}$$

The maximum is taken over all B with execution time t' and at most q' queries to the oracle.

The following theorem means that she cannot compute MAC values of non-captured nodes from MAC values of captured nodes if the message authentication code is secure.

Theorem 1. Let $\mathsf{MAC} : Key(\mathsf{MAC}) \times Dom(\mathsf{MAC}) \longrightarrow \{0,1\}^k$ be a family of functions, and let $q, t, q', t' \geq 1$ be integers. Let M be the proposed scheme. Then

$$\mathsf{Adv}^{\mathsf{cna}}_{\mathsf{M}}(q', t') \leq \mathsf{Adv}^{\mathsf{cma}}_{\mathsf{MAC}}(q, t)$$

where $q \leq q' \cdot (1+n)$ and $t = t' + O(k)$.

Proof) Let B be an attacker breaking M. We construct an attacker A_B breaking MAC. Consider the following experiment.

```
Attacker $A_B^{\mathsf{MAC}(K,\cdot)}$
    Generate $G = (N, E)$
    $G$ is given to $B$
    Run attacker $B$, replying to its oracle queries as follows:
    While $B$ asks a query $u'$ to the oracle do
        Generate $str \in \{0,1\}^{k(1+n)}$ for a expected degree $n$ of a node such that
            1. $(u', \mathsf{MAC}(K, u'))$ is in $str$,
            2. for each $\langle u', v' \rangle \in E$, $(u', v', \mathsf{MAC}(K, u'||v'))$ is in $str$, and
            3. for each $\langle v', u' \rangle \in E$, $(v', u', \mathsf{MAC}(K, v'||u'))$ is in $str$.
        Return $str$ to $B$ as an answer
    EndDo
    $B$ stops and outputs $(u, \sigma)$ or $((u, v, \sigma)$ or $(v, u, \sigma))$
    $A$ returns the output of $B$
```

Here A_B is running B and provides answers to B's oracle queries. When B asks a node capture query u', attacker A_B needs to return str which is the memory information of node u'. B returns (u, σ) or $((u, v, \sigma)$ or $(v, u, \sigma))$ which is a valid forgery of MAC. So, we have

$$\mathsf{Succ}^{\mathsf{cna}}_{\mathsf{M}}(B) \leq \mathsf{Succ}^{\mathsf{cma}}_{\mathsf{MAC}}(A_B). \tag{1}$$

Inequality of the theorem is obtained as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{cna}}_{\mathsf{M}}(q', t') &= \underset{B}{max}\{\mathsf{Succ}^{\mathsf{cna}}_{\mathsf{M}}(B)\} \\
&\leq \underset{B}{max}\{\mathsf{Succ}^{\mathsf{cma}}_{\mathsf{MAC}}(A_B)\} \\
&\leq \underset{A}{max}\{\mathsf{Succ}^{\mathsf{cma}}_{\mathsf{MAC}}(A)\} \\
&= \mathsf{Adv}^{\mathsf{cma}}_{\mathsf{MAC}}(q, t).
\end{aligned}$$

The maximum, in the case of B, is taken over all adversaries whose resources are q', t'. In the second line, we apply Inequality (1). ■

5 Conclusion

We presented a new key scheme for large-scale distributed sensor networks. Our scheme has the following properties. First, compared to LEAP, our scheme is significantly more efficient and secure. Second, we prove the security of our scheme after key setup. Finally, our scheme supports large-scale networks because performance overhead in our scheme is independent of network size. Unlike other scheme [4, 6–9], both our scheme and LEAP have perfect resilience against node capture after key setup. However, since they are weak during key setup, this paper was focused on designing an efficient key establishment scheme.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks”, In Proceedings of the IEEE Communications Magazine, Vol. 40, No. 8, pp. 102-114, August 2002.
2. M. Bellare, J. Kilian, and P. Rogaway, “The security of the cipher block chaining message authentication code”, Journal of Computer and System Sciences, Vol. 61, No. 3, pp. 362-399, December 2000.
3. D. W. Carman, P. S. Kruus, and B. J. Matt, “Constraints and approaches for distributed sensor network security”, NAI Labs Technical Report 00-010, September 2000.
4. H. Chan, A. Perrig, and D. Song, “Random key predistribution schemes for sensor networks”, In Proceedings of the 2003 IEEE Symposium on Security and Privacy, pp. 197-213, May 2003.
5. Crossbow technology inc. URL: http://www.xbow.com.
6. W. Du, J. Deng, Y. S. Han, S. Chen, and P.K. Varshney, “A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge”, In Proceedings of the IEEE INFOCOM ‘04, pp. 586-597, March 2004.
7. W. Du, J. Deng, Y. S. Han, P.K. Varshney, J. Katz, and A. Khalili, “A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks”, In Proceedings of the ACM Transactions on Information and System Security, pp. 228-258, August 2005.
8. L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks”, In Proceedings of the 9th ACM conference on Computer and communications security, pp. 41-47, November 2002.
9. D. Liu, P. Ning, and R. Li, “Establishing Pairwise Keys in Distributed Sensor Networks”, In Proceedings of the ACM Transactions on Information and System Security, pp. 41-77, February 2005.
10. A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar, “SPINS: Security protocols for sensor networks”, In Proceedings of the 7th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking, pp. 189-199, July 2001.
11. S. Zhu, S. Setia, and S. Jajodia. “LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks”, In Proceedings of the Tenth ACM conference on Computer and Communications Security, pp. 62-72, October 2003.
12. S. Zhu, S. Setia, and S. Jajodia. The technical report about LEAP, URL: http://www.cse.psu.edu/ $\sim$ szhu/research.htm, August 2004.