A Time-Based Key Management Protocol for Wireless Sensor Networks*

Jiyong Jang[1], Taekyoung Kwon[2], and Jooseok Song[1]

[1] Department of Computer Science, Yonsei University
{souljang, jssong}@emerald.yonsei.ac.kr.
[2] Department of Computer Engineering, Sejong University
tkwon@sejong.ac.kr.

Abstract. It is not easy to achieve secure key establishment in wireless sensor networks without public key cryptography. Many key management protocols have been proposed for the purpose. Among them, LEAP is a simple and elegant protocol that establishes multi-level keys in an efficient way, but its security mainly relies on that of a single initialization key. Though it is assumed that the initial deployment phase is secure and the key is erased from sensor nodes after the initialization in LEAP, the assumption could not be viable for two reasons. First, the same key should be used again for node addition after the initialization phase whereas the new node can be captured before removing the key. Second, the initial deployment of dense networks may not take short as LEAP expected in many cases. This paper rethinks the security of LEAP and proposes a more secure scheme with a new notion of probabilistic time intervals. Rather we localize the impact of key compromise within the time intervals.

Keywords: Wireless Sensor Network, Security Protocol.

1 Introduction

Wireless sensor networks (WSNs) are dense wireless networks of sensor nodes which are constrained in their computation, communication, and storage capabilities. They are expected to play an important role in many applications such as environment monitoring, building management, health care, and military operation. Since they are deployed in unattended or even hostile environments, security mechanisms are required for various mission critical applications [1,2]. However, it is widely recognized that securing WSNs is quite challenging due to the limited features of sensor nodes.

Many key management schemes have been proposed to make secure links in WSNs. A probabilistic key pre-distribution scheme is proposed in [1]. In this scheme, a randomly chosen set of keys from a large key pool is assigned to each

* This work was supported by grant No. R01-2006-000-10614-0[1] and No. R01-2005-000-11261-0[2] from the Basic Research Program of the Korea Science & Engineering Foundation.

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 314–328, 2007.

sensor node before node deployment. And then, two sensor nodes can share at least a common key with a certain probability. This scheme is improved in [3]. Two sensor nodes are required to share at least q secret keys to establish a pair-wise key. A random pair-wise key scheme is also introduced in [3] to provide perfect security against node capture. [4,5] use a threshold-based technique: If the number of compromised nodes does not exceed a threshold value, the rest of network is not affected by compromised ones. Recently researchers have suggested to utilize the expected location of sensor nodes after node deployment to improve the security and scalability of key establishment schemes [6,7,8]. However, it is limited to take advantage of the knowledge of locations since it is very difficult to guarantee the exact positions of sensor nodes.

Lately an efficient key management protocol called LEAP (Localized Encryption and Authentication Protocol) [9] has been proposed by Zhu, Setia, and Jajodia in order for supporting secure key establishment and in-network processing for large-scale WSNs by establishing four types of keys such as individual key, group key, cluster key, and pairwise shared key.

Most of the key management protocols including LEAP assume that an adversary may attack sensor networks after the initial key establishment phase, but the assumption could be incorrect while considering node addition phases in a hostile environment. Security of LEAP mainly depends upon that of the initialization key which is erased from sensor nodes after the initialization phase. However, the same key should be used again for node addition after that phase while the new node can be captured before removing the initialization key. In this paper, we rethink the security of LEAP and introduce a time-based key management protocol which improves security with a new notion of probabilistic time intervals.

This paper is structured as follows: We describe existing key management protocols in Section 2, and then rethink the LEAP protocol in terms of security in Section 3. We propose a time-based key management protocol in Section 4, and then analyze its performance and security in Section 5. We conclude this paper in Section 6.

2 Related Works

2.1 Key Management Protocols in WSNs

To provide secure communications in WSNs, sensor nodes first need to set up pair-wise keys with each other. There are generally three types of key agreement schemes: the trusted-server scheme, the self-enforcing scheme, and the key pre-distribution scheme [6]. The trusted-server scheme assumes that there is a trusted server for key establishment between nodes. However, this is not suitable for distributed sensor networks since it is usually hard to construct a trusted server. The self-enforcing scheme uses asymmetric cryptography, such as a public key certificate. However, a public key algorithm is not suitable for sensor networks because of limited power and computation resources of tiny sensor nodes. In key pre-distribution schemes, keying materials are pre-loaded into sensor nodes prior

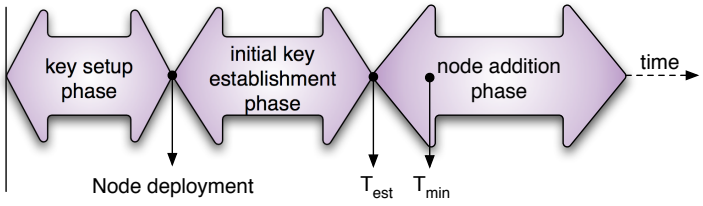

Fig. 1. 3 phases of key pre-distribution scheme

to the deployment. If we can utilize the location information of sensor nodes, we could make the scheme more efficient.

A key pre-distribution scheme consists of three phases in general: a key setup phase prior to deployment, an initial key establishment phase, and a node addition phase. We refer readers to Figure 1. During a key setup phase, a center generates keying materials which will be used to make a secure link, and then pre-loads some keying materials into sensor nodes prior to node deployment. After each sensor node discovers neighbor nodes sharing a common keying material, sensor nodes are able to establish pair-wise keys with each other during an initial key establishment phase. Sensor nodes which do not share any common keying materials, but are within the wireless communication range, can establish a path-key via a proxy node that already has pair-wise keys with both nodes. During the node addition phase, some additional nodes are deployed in sensor networks for several reasons, such as maintenance, replacement, routing, and so on. Now, we summarize two basic key pre-distribution schemes.

There are a lot of key pre-distribution schemes. At first, we can easily think about two naive solutions. One solution is to use a single master key in a whole network. Any pair of nodes can establish pair-wise keys using this master key. However, if one node is compromised, the whole network can be threatened by an attacker. The other solution is to assign a unique pair-wise key to every pair of nodes. Each sensor is required to store $N - 1$ pair-wise keys so that whole sensor nodes are required to store $N(N - 1)/2$ secret keys. Since compromising one node does not affect the rest of network, this scheme is perfectly resilient to node compromise. However, this scheme is not suitable for a large sensor network because of a limited memory constraint of sensor nodes.

2.2 EG Scheme

Overview. Eschenauer and Gligor proposed a probabilistic key pre-distribution scheme in [1]. This scheme relies on probabilistic key sharing between sensor nodes. At first, a center generates a large key pool P, and then randomly select k keys from pool P for each sensor node without replacement. k keys form a key ring of a sensor node, and the key ring is pre-loaded into a node prior to node deployment. As a result, two nodes share at least one key with a certain probability. After node deployment, each sensor node discovers its neighbors which share a common key in wireless communication range to establish a

pair-wise key. Such shared-key discovery phase establishes the topology of sensor network and path-key establishment phase reinforces a key connectivity of sensor network.

2.3 LEAP

Overview. LEAP is a cryptographic protocol allowing secure key establishment for WSNs [9]. In LEAP, it is assumed that sensor nodes are not mobile and every node has enough space to store hundreds of bytes of keying materials. It is also assumed that an adversary can eavesdrop on all traffic and extract all the information from the compromised node. LEAP provides confidentiality and authentication for secure communications in WSNs. LEAP is designed to support in-network processing such as data aggregation and passive participation. In-network processing can remove transfer of redundant messages so that energy consumption can be reduced. LEAP offers multiple keying mechanisms resulting from that different types of messages having different security requirements. LEAP also provides one-way key chain based authentication for inter-node traffic.

Establishment of Four Types of Keys. LEAP offers four types of keys to each sensor node - an individual key shared with the center, a pairwise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a group key shared by all the nodes in the network - and we summarize how to establish those keys below.

- **Individual Key:** Each node has a unique key shared with the center. This key is used for secure communication between a sensor node and the center. The individual key is generated as $K_u^m = f_{K^m}(ID_u)$ where f is a pseudo-random function and K^m is a master key known only to the center and ID_u is the id of a node u. This generated key is pre-loaded into a sensor node prior to node deployment.

- **Pairwise Key:** Every node has a pairwise shared key with its immediate neighbors respectively. Pairwise shared keys are used for secure distribution of cluster keys to its direct neighbor nodes and secure transmission of data. The center generates an initial key K_I and a node u computes a master key $K_u = f_{K_I}(ID_u)$. During neighbor discovery stage, node u broadcasts a HELLO message within its id and waits for response from neighbor v. The response message from node v contains id of v and message authentication code (MAC) for verifying node v's identity. And then, node u is able to authenticate node v since it can compute MAC value with the master key K_v which is derived as $K_v = f_{K_I}(ID_v)$.

$$u \rightarrow * : ID_u, Nonce_u$$
$$v \rightarrow u : ID_v, MAC_{K_v}(Nonce_u | ID_v)$$

  After authentication, node u computes a pairwise key with v as $K_{uv} = f_{K_v}(ID_u)$. Node v can also derive K_{uv} in the same way.

– **Cluster Key:** Each node has a common shared key with all its neighbors for supporting in-network processing. This key is used for encrypting or authenticating local broadcast messages. Node u first generates a random cluster key K_u^c and then encrypts the key with the pairwise shared key with each neighbor respectively and sends the encrypted key to each neighbor v_i. Each neighbor node decrypts the received message to get the cluster key K_u^c.

$$u \text{ generates cluster key } K_u^c$$
$$u \rightarrow v_i : (K_u^c)_{K_{uv_i}}$$

When one of the neighbor nodes is compromised, node u needs to generate a new cluster key and transmits it to the remaining neighbor nodes.

– **Group Key:** Every node has a globally shared key in the whole network. Group key is used for encrypting broadcast messages by the center. We can simply pre-load a group key into each node. We need to update the key periodically or when a compromised node is detected, that is, group rekeying problem must be considered.

– **Multi-hop Pairwise Shared Key:** A node has a pairwise shared key between a node and the aggregation node. This key is used by a node for transmitting data to an aggregation node which is multiple hops away. A node u first broadcasts a message with its id ID_u and id of the cluster head ID_c. The nodes which already have a pairwise shared key with both the node u and the cluster head c send reply messages to the node u. Now the intermediate nodes become the proxies. To establish a pairwise key S with cluster head c, node u splits S into m shares, such that $S = sk_1 \bigoplus sk_2 \bigoplus ... \bigoplus sk_m$, and then transmits each sk_i to the cluster head c via proxy v_i.

$$u \rightarrow v_i : (sk_i)_{K_{uv_i}}, f_{sk_i}(0)$$
$$v_i \rightarrow c : (sk_i)_{K_{v_ic}}, f_{sk_i}(0)$$

where $f_{sk_i}(0)$ is the verification key of key sk_i since the cluster head c can verify the validation of sk_i. After the cluster head c receives all shares, it restores a pairwise shared key S.

LEAP assumes that T_{min}, the time interval for an attacker to compromise a node, is larger than T_{est}, the time for a newly deployed node to complete neighbor discovery stage, as depicted in Figure 1. In LEAP, all nodes erase an initial key K_I and all the neighbors' master keys after time T_{min}. Therefore, an attacker compromising a node after T_{min} can obtain only the keying materials of the compromised node, not those of other nodes. Therefore, the affected fraction of the network due to node compromise can be localized. When a node compromise is detected, its neighbor nodes just erase the keys shared with the compromised node.

Local Broadcast Authentication. LEAP supports one-way key chain based authentication for local broadcast messages. Each node generates a one-way key

chain composed of keys called AUTH key, and sends the commitment (the first key) of key chain to its neighbors encrypting with each pairwise key. When a node transmits a message, it attaches the next AUTH key in the key chain to the message. The AUTH keys are disclosed in a reverse order. The commitment and received AUTH key allow a receiving node to verify the received message. Unlike μTESLA [11] which uses delayed key disclosure and requires time synchronization between neighboring nodes, this mechanism can provide immediate authentication.

3 Rethinking LEAP and Its Security

Most of the key agreement schemes assume that sensor networks are relatively secure against attacks during the initial key establishment phase and an adversary may capture sensor nodes to compromise the network after the phase. The LEAP protocol also has a similar assumption like that T_{est} is smaller than T_{min}. However, this assumption is often not true. For example, packet losses due to reasons such as narrow bandwidth or bad channel condition of sensor networks may happen while sensor nodes transmit data to each other during initial key establishment phase. This can cause several retransmissions of packets so that the time for sensor nodes to establish pair-wise keys each other, T_{est}, may take longer than expected, even T_{min}. Also T_{est} may be on the order of tens of minutes in certain deployment schemes.

If sensor nodes are scattered from airplanes or helicopters, then the nodes may settle sparsely and need enough time to set up the network and establish pairwise keys. During this time, an adversary can capture a sensor node and get an initial key K_I. Moreover some researches [12] show that it takes less than 1 minute to dump all of the EEPROM, program Flash, and a chip's SRAM. An initial key K_I can be disclosed if it only takes on the order of seconds to compromise a node. LEAP also assumes that the node moves K_I from non-volatile memory into volatile memory to make the scheme more secure. However, this assumption is not true since both RAM and flash are accessible to an adversary [12]. In above cases, an adversary is able to get an initial key K_I, and then inject erroneous information or add new nodes at her pleasure.

Moreover, in case of LEAP, a newly deployed node which is added to the network after the initial key establishment phase will carry the initial key K_I and may be captured in an hostile environment. Thus, as for node addition in LEAP, an initial key K_I should never be used after the initial time T_{min} without permission even for the legitimate new nodes since an adversary is able to capture a node in the initial T_{min} and find out an initial key K_I within T_{min} afterward.

4 A Time-Based Key Management Protocol

4.1 A Time-Based Deployment Model

As we have already mentioned in the previous section, the time for sensor nodes to establish pair-wise keys each other, T_{est}, may take longer than the time

interval for an adversary to compromise a node, T_{min}. If an initial key K_I is disclosed within T_{min} time, then the whole sensor network is threatened by an attacker. Even though an initial key K_I is disclosed by an attacker, the portion of network compromised must be minimized. For that reason we split the time domain to disperse the damage resulting from the disclosure of an initial key K_I. Now we introduce more secure key pre-distribution protocol with time-based multiple K_I. Selection of K_I with probabilistic time intervals is as follows:

Key Setup Phase

- First a center generates a pool of P initial keys and divides whole lifetime of sensor network into P time slots.
- A center assigns an initial key to each time slot.
- The initial key of the deployment time slot and m master keys of randomly-chosen time slots are pre-loaded into sensor nodes prior to deployment. When the deployment time slot is T_i, sensor nodes stores an initial key K_{Ii} and m randomly-chosen master keys.

Initial Key Establishment Phase

- According to the original LEAP protocol, an initial key establishment phase means the first time slot T_1.
- Since all sensor nodes deployed at an initial key establishment phase contains the initial key K_{I1}, they can establish pair-wise keys using K_{I1}.
- After a node u computes a master key $K_{u1} = f_{K_{I1}}(ID_u)$, node u broadcasts a HELLO message within its ID and then waits for a response from neighbor node v. Node v sends node u a response message including its ID and MAC.

$$u \rightarrow * : ID_u, Nonce_u$$
$$v \rightarrow u : ID_v, MAC_{K_{v1}}(Nonce_u | ID_v)$$

- Now both u and v can compute a pair-wise key $K_{uv} = f_{K_{v1}}(ID_u)$.

Node Addition Phase

- During a node addition phase, newly deployed nodes first discover neighbor nodes which share common keying materials.
- Sensor nodes are able to generate pair-wise keys with other nodes which are deployed at the same time slot using the same initial key.
- And then, sensor nodes can establish pair-wise keys with other nodes which are deployed at different time slots, but have the master key derived from the current initial key.

  Let u and v be a newly deployed node at time slot T_k and a pre-deployed node respectively. If a node v has the master key K_{vk} derived from the

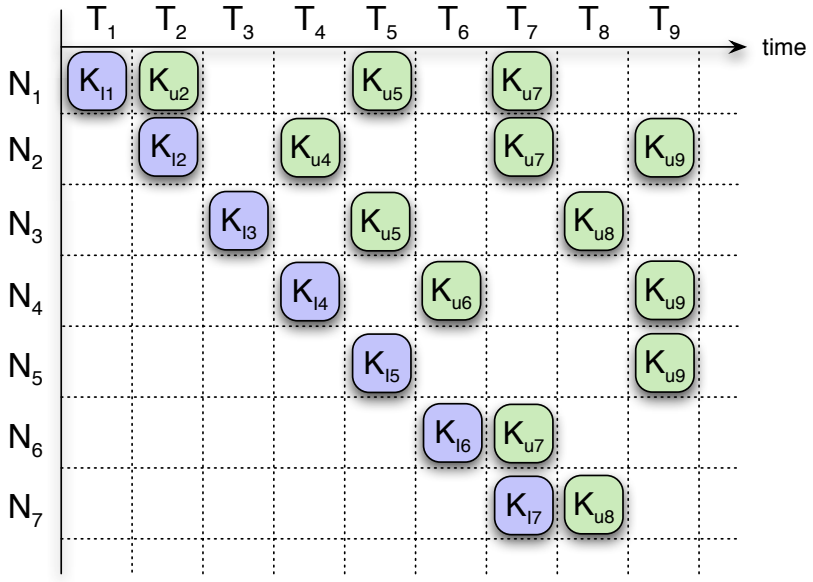

Fig. 2. Example of selecting K_I with probabilistic time intervals

current initial key K_{Ik}, both node u and v can compute a pair-wise key $K_{uv} = f_{K_{vk}}(ID_u)$ because node u is also able to generate a master key of v, K_{vk}, using the current initial key K_{Ik} and ID of v.

- After all, during the deployment time slot, sensor nodes can establish pair-wise keys with each other by using the initial key. For the other m time slots, sensor nodes are able to establish a secure link with other nodes by using an appropriate one of the m master keys.
- After that, a pair of sensor nodes that do not share a keying material but are in wireless communication range can establish path-keys via proxy nodes.

4.2 Example of Time-Based Deployment Model

Readers are referred to Figure 2. T_n and N_n represent each time slot and group of nodes deployed at T_n time slot, respectively. As shown in Figure 2, nodes of group N_1 store the initial key K_{I1} of time slot T_1 and m master keys of randomly-chosen time slots including K_{u2}, K_{u5} and K_{u7}. Since, according to original LEAP protocol, master key is derived from an initial key and node ID, $K_{u2} = f_{K_{I2}}(NodeID)$. All nodes of group N_1 are able to establish pair-wise keys each other using the initial key K_{I1} during time slot T_1. Then, they are able to establish a secure communication with nodes of group N_2 using the master key K_{u2} during time slot T_2. Note that all nodes of group N_2 can also derive the master

key K_{u2} from the initial key K_{I2}. Similarly, they can also generate pair-wise keys with nodes of group N_5 and N_7 using master keys K_{u5} and K_{u7}, respectively.

4.3 Practical Application of Time-Based Deployment Model

Location-based key management protocols are very efficient methods in terms of key connectivity and storage overhead. However, location information of sensor nodes is crucial since these schemes utilize the locations of sensor nodes. Even though the exact deployment locations of sensor nodes are not necessarily in these schemes, we must know the dense spots of sensor nodes prior to node deployment. Moreover, in case that we divide the whole region into many small areas, these schemes can provide higher resilience, but have a great difficulty in deploying sensor nodes. On the other hand, in case that we have the large size of areas, these schemes would be more vulnerable to attacks, but have no difficulty in deploying sensor nodes. That is, these schemes have the tradeoff between the security and the easy deployment.

On the contrary, location information of sensor nodes does not required to employ a time-based deployment model so that additional sensor nodes are easily added on WSN without the restriction of deployment points in our scheme. Therefore, our scheme is more beneficial to node deployment. Furthermore, as we described above, a time-based deployment model divides time into many small parts; thus, the effect of node capture attack can be localized into a small part. That is, we are able to have both high resilience and no difficulty in deploying sensor nodes by taking advantage of a time-based deployment model.

A time-based deployment model can be employed in a more practical way. In this model, every time slot appears continually, which means node addition must occur periodically. In fact, however, after initial deployment phase, additional nodes will be added to WSN irregularly. Therefore, we suggest the practical application of time-based deployment model as depicted in Figure 3. During a normal operation phase, WSN performs a normal operation such as sensing and transferring data without node additions. To deploy additional nodes, a center broadcasts an authenticated packet which notices pre-deployed sensor nodes to prepare node additions. The packet will contain information such as when the following time slot starts and for how many time slots node addition lasts. We assume that the length of a time slot is very short, in that sensor nodes just need to time to exchange keying materials and generate pair-wise keys during a time slot.

In a practical application of time-based deployment model, the beginning of new time slots means the occurrence of node additions so that the number of time slots is approximately equal to the number of occurrences of node addition. The number of time slots also means the size of key pool. Therefore, the size of key pool is influenced by the frequency of node additions. If the additional sensor nodes are deployed for the purpose of complementing sensor networks, the frequency of node additions is not necessarily high; thus, the size of key pool could be small.

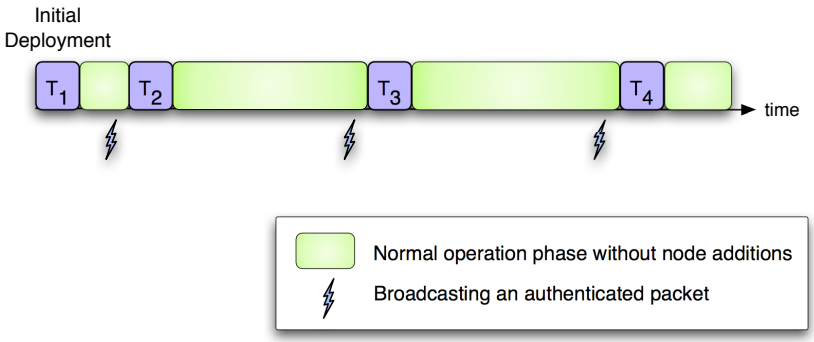

Fig. 3. Practical application of time-based deployment model

5 Performance and Security Analysis

5.1 Performance Analysis

Key Connectivity. When sensor nodes are deployed at a specific time slot, pre-deployed sensor nodes must have the master key, which is derived from the initial key of that time slot, to establish pair-wise keys each other. For example, N_7 group of sensor nodes are deployed at time slot T_7. To make secure links, N_1-N_6 group of sensor nodes must have the master keys which is derived from the initial key K_{I7}. At first, consider N_1's probability of sharing a keying material with N_7. The probability is [the number of cases where $m-1$ master keys are chosen from key pool except for K_{I1} and K_{I7}]/[the number of cases where m master keys are chosen from key pool except for K_{I1}]. That is, $\binom{P-2}{m-1}/\binom{P-1}{m} = \frac{m}{P-1}$. Since the master keys of every time slot would be chosen with the same probability, N_1's probability of sharing keying materials with other prospective sensor nodes(N_2-N_P) is $\frac{m}{P-1}$. Now, N_i's probability of sharing keying materials with prospective sensor nodes, $p_{pros}(i)$, can be calculated as:

$$p_{pros}(i) = \frac{\binom{P-i-1}{m-1}}{\binom{P-i}{m}} = \frac{m}{P-i} \tag{1}$$

as long as $(P-i)$ is greater than m; otherwise, $p_{pros}(i) = 1$. Figure 4 describes Eq. 1 for various values of m where the size of key pool P is 500. Although the size of 500 seems small for the key pool, it will be enough to operate the sensor network. (Refer to section 4.3.) After $(P-i)$ becomes smaller than m, the master keys of all remaining time slots will be chosen so that the probability $p_{pros}(i)$ is 1. As shown in the figure, the probability p_{pros} becomes higher as the index of time slot i increases, in that the size of key pool from which master keys are chosen becomes smaller. Also, the more master keys sensor nodes have, the higher the probability becomes.

Now we are able to compute $p_{time}(t)$, which means the probability that N_t group of sensor nodes share keying materials with pre-deployed sensor nodes and

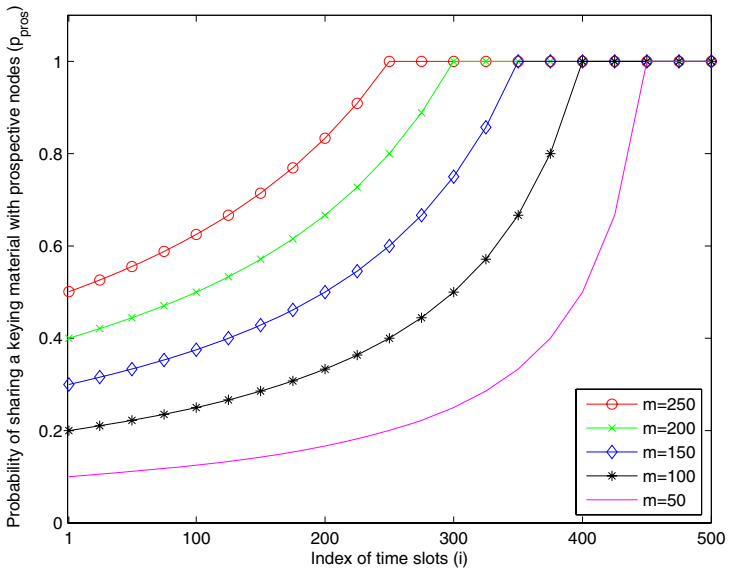

Fig. 4. [p_{pros}] The probability that N_i shares keying materials with prospective sensor nodes

N_t themselves at time slot T_t, using p_{pros}. If we assume that sensor nodes are uniformly distributed at every time slot, p_{time} is:

$$p_{time}(t) = (\sum_{i=1}^{t-1} p_{pros}(i) + 1)/t \tag{2}$$

which means the average value of the probabilities of sharing keying materials with each pre-deployed group of sensor nodes. In Eq. 2, 1 means that all sensor nodes deployed at the same time slot can make secure links. Figure 5 draws Eq. 2 for various values of m where the size of key pool P is 500. At time slot T_1, all sensor nodes have the same initial key so that the probability is 1. As time passes, sensor nodes would be added to the sensor network and the probability p_{time} drops, in that p_{pros} is much smaller than 1. After the probability p_{time} nearly decreases to the minimum value of p_{pros}, p_{time} becomes higher since p_{pros} increases.

The direct key connectivity C that a sensor node is able to generate pair-wise keys with at least one of the immediate neighbor nodes (i.e., 1-hop connectivity) is derived as follows:

$$C = 1 - (1 - p_{time})^d \tag{3}$$

where d means the average number of neighbor nodes. To calculate the key connectivity and show the effect of neighbor nodes on the key connectivity, we

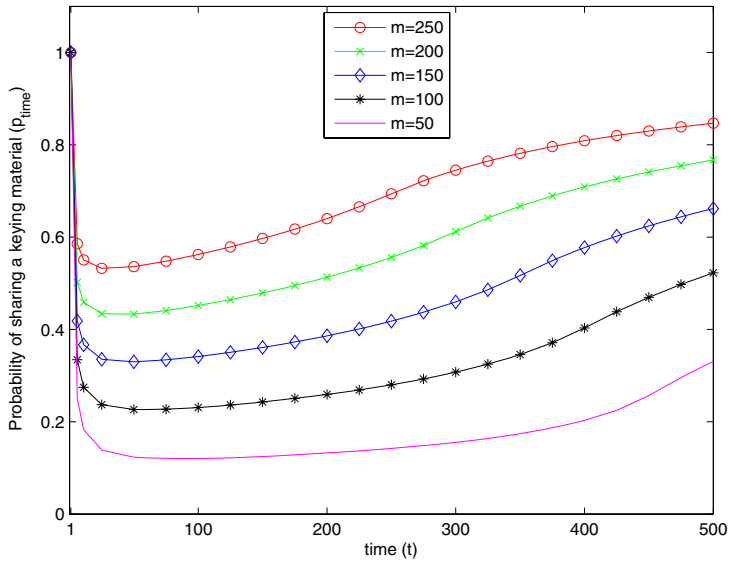

Fig. 5. [p_{time}] The probability that N_t shares keying materials with pre-deployed sensor nodes and the nodes being deployed in the same time interval

choose the lowest value of p_{time} from Figure 5. Figure 6 describes Eq. 3 for various values of m where the size of key pool P is 500 and p_{time} has the minimum value. As shown in Figure 6, while the number of keys stored in a sensor node remains the same, the key connectivity goes high as the number of neighbor nodes increases.

Storage Overhead. As for the storage requirement, we can evaluate that our scheme needs a reasonable cost in modern sensor nodes such as MICA-Z and Telos. Sensor nodes choosing about 100 keys from key pool of size 500 are able to share a keying material with at least one of the 10 neighbor nodes with more than 0.9 probability (refer to Figure 6). Note that this requirement only corresponds to the node addition phase, while the connection probability is 1, saying, deterministic at the initial deployment. Also note that we only consider the direct key connectivity (1-hop connectivity) as for the probability in Figure 6. The remaining unconnectivity can be resolved by the help of other (proxy) neighbor nodes easily. When the size of a key is 64 bits, a center and a sensor node need approximately 4KB and 0.8KB memory space, respectively. The number of keys saved in memory will decrease as old keys can be discarded. For strengthening the security of WSNs with regard to node capture, we believe the initial requirements of 0.8KB are reasonable for the modern sensor nodes such as MICA-Z having 128KB program memory, 4KB runtime memory, and 512KB external memory [13].

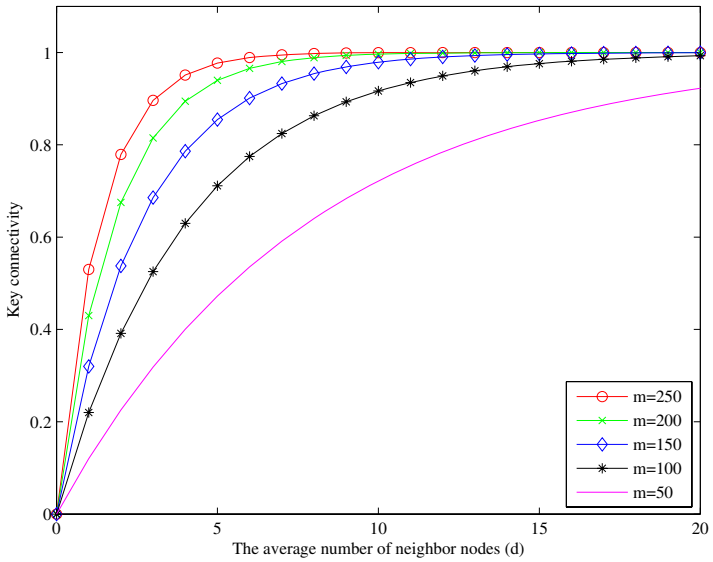

Fig. 6. Key connectivity that a sensor node can make a secure link with at least one of the immediate neighbor nodes

5.2 Security Analysis

When node compromise is detected, the keying materials which are associated with the compromised node must be revoked to prevent an adversary from attacking the rest of network using information extracted from compromised node. Since detection of node compromise is not easy, the additional portion of network that an adversary can compromise using the keying materials obtained from x captured nodes represents the resilience of schemes. In other words, the resilience of schemes means the survivability of network against node compromise.

Since an initial key K_I is removed from a node after T_{est} in original LEAP protocol, an attacker can use only pair-wise keys and cluster keys even if she succeeded in capturing a node. An attacker is not able to use the compromised node in other areas so that the affected fraction of network due to node capture is localized. Since the portion of network compromised is localized, wormhole attack or sinkhole attack [14] can be protected. However, if an adversary succeeds in capturing a node before T_{est}, she is able to get an initial key K_I so that the whole network can be compromised.

The damage resulting from a disclosure of an initial key K_I can be minimized by using selection of K_I with probabilistic time intervals scheme. Even if an adversary is able to get an initial key K_{Ia} at time slot T_a, only the nodes deployed at time slot T_a, not whole network, are compromised. Even if an attacker knows an initial key K_{Ia}, she cannot retrieve the previous initial keys so that she is never able to get an information from transmitted data before time slot T_a. Therefore, the selection of K_I with probabilistic time intervals scheme provides backward

confidentiality. As only master keys, not initial keys, are stored in sensor nodes for the other m time slots, this scheme also provides forward confidentiality.

5.3 Comparison

Table 1 shows key connectivity, key storage overhead and resilience of EG scheme, LEAP, and our scheme where N is the size of the sensor network, d is the average number of neighbor nodes, and t is the index of time slots. In EG scheme, m keys out of key pool P are chosen for each sensor node.

Table 1. Comparison with EG scheme and original LEAP

		EG scheme [1]	LEAP [9]	our scheme
Key connectivity		$p_1 = 1 - \frac{((P-m)!)^2}{P!(P-2m)!}$	1	$(\sum_{i=1}^{t-1} \frac{m}{P-i} + 1)/t$
Initial key storage overhead		m	1	$m+1$
Compromised network due to node capture	after T_{est}	$p_1 \cdot (N-1)$	d	d
	before T_{est}	$p_1 \cdot (N-1)$	*whole network*	*one group of* N_i
Resilience against wormhole attack or sinkhole attack		X	O	O
Forward confidentiality		X	X	O
Backward confidentiality		X	X	O

Because of the high key connectivity between the nodes deployed at the same time slot, key connectivity of our scheme is higher than that of EG scheme while the initial key storage overhead remains unchanged. Since we select multiple K_I with probabilistic time intervals, LEAP shows the better performance than ours in terms of key connectivity and storage overhead. However, our scheme is able to minimize the impact of node capture attack even though an attacker succeeds the node capture attack before T_{est}. Note that if an adversary can compromise a sensor node before T_{est}, K_I is disclosed so that the whole network is compromised in LEAP. Ours as well as LEAP prevent an adversary from launching wormhole attack or sinkhole attack because a node knows all its neighbors after neighbor discovery. Since an attacker cannot derive previous initial keys from the current initial key or next initial keys from master keys, our scheme provides backward confidentiality and forward confidentiality.

6 Conclusions

The multiple keying mechanism of LEAP satisfies the multiple usages of sensor networks, but we have found that LEAP actually missed a possible disclosure of an initial key K_I. Its security mainly relies on that of an initialization key while the

initial deployment phase is assumed secure and the key is erased from sensor nodes after the initialization phase. However, the same key should be used again for node addition after that phase while the new node can be captured before removing the initialization key. And the assumption of security in the initial deployment phase is not viable in many cases since the initial deployment of dense networks may not take short as LEAP expected. This paper rethinks the security of LEAP and proposes a more secure scheme with a new notion of probabilistic time intervals. Rather we localize the impact of K_I disclosure within the time intervals.

References

1. L. Eschenauer and V. D. Gligor, "A Key-management Scheme for Distributed Sensor Networks," in *Proc. of the 9th ACM Conference on Computer and Communication Security (CCS'02)*, pp. 41-47, Nov. 2002.
2. A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," Communications of the ACM, vol. 47, no. 6, pp. 53-57, June 2004.
3. H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," in *IEEE Symposium on Research in Security and Privacy*, pp. 197-213, 2003.
4. W. Du, J. Deng, Y.S. Han, and P. Varshney, "A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003.
5. D. Liu, and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security(CCS'03)*, pp. 52-61, October 2003.
6. W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge," in *IEEE Transactions on dependable and secure computing*, vol. 3, no. 1, pp. 62-77, Feb. 2006.
7. D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-Aware Key Management Scheme for Wireless Sensor Networks," in *Proc. of ACM Workshop Security of Ad Hoc and Sensor Networks (SASN'04)*, pp. 29-42, Oct. 2004.
8. J. Lee, T. Kwon, and J. Song, "Location-aware key management using multi-layer grids for wireless sensor networks," Applied Cryptography and Network Security, Lecture Notes in Computer Science, Vol. 3989, Springer-Verlag, pp. 390-404, 2006.
9. S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proc. of the 10th ACM Conference on Computer and Communication Security (CCS'03)*, pp. 62-72, November 2003.
10. R. Blom, "An Optimal Class of Symmetric Key Generation Systems," in *Advances in Cryptology: Proc. EUROCRYPT '84*, pp. 335-338, 1985
11. A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar, "SPINS: Security protocols for sensor networks", in *Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 189-199, July 2001.
12. C. Hartung, J. Balasalle, and R. Han, "Node Compromise in Sensor Networks: The Need for Secure Systems," Technical Report CU-CS-990-05, University of Colorado at Boulder, Jan. 2005.
13. Crossbow Technology (http://www.xbow.com)
14. C. Karlof and D. Wagner, "Secure Routing in Sensor Networks: Attacks and Countermeasures," in *Proc. of the 1st IEEE Workshop on Sensor Network Protocols and Applications*, May 2003.