A Location-ID Sensitive Key Establishment Scheme in Static Wireless Sensor Networks

Li Chen
Graduate Institute of Networking and Multimedia, National Taiwan University
No.4, Sec. 1, Roosevelt Rd
Taipei, Taiwan
neo@fractal.ee.ntu.edu.tw

Chia-Chang Hsu
Department of Electrical Engineering, National Taiwan University
No.4, Sec. 1, Roosevelt Rd
Taipei, Taiwan
oberon@fractal.ee.ntu.edu.tw

Chin-Laung Lei
Department of Electrical Engineering, National Taiwan University
No.4, Sec. 1, Roosevelt Rd
Taipei, Taiwan
lei@cc.ee.ntu.edu.tw

ABSTRACT

Sensor networks are usually consist of thousands of resource-limited nodes and are deployed in a designated area without any fixed infrastructure. While the establishment of the pairwise keys between any pair of adjacent nodes to build a secure link remains the main concern in the design of key management protocols, malicious attacks aim at routing information, exhaust node's resource, and compromised secrets can misdirect the data flow or denial the network service with relatively small effort. Many mission-critic sensor network applications demand an effective, light, and flexible algorithm yet robust under attacks.

Based on the LEAP+ scheme, we propose an improved LEAP+ by adding location information into the key establishment phase. By identifying the correctness of the id-location pair, our scheme effectively limits the Sybil attack and mitigates the damage of HELLO flood attack and node cloning attack. We furthermore propose an authentication phase in our scheme to defend possible replay attacks. The analysis shows that our scheme is more robust than LEAP+ with only minor increase of computation overhead.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General – *Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Wireless Communication*;

General Terms

Performance, Security

Keywords

location-based key establishment, HELLO flood attack, node cloning, sybil attack.

This work is supported in part by the Taiwan Information Security Center (TWISC), the National Science Council under the Grants No. NSC 97-2219-E-001-001, and by the international collaboration for advancing security technology (iCAST), National Science Council under the Grants No. NSC 96-3114-P-001-002-Y.

1. INTRODUCTION

Sensor networks are often deployed in various situations where different security requirement must be satisfied for different situation. For example, consider a patient monitoring system that sensors are placed in situ the patient, the data confidentiality must be hold since we don't want to leak out patient's private data, and for the sensor networks that uses in hostile environment such as battlefield, they should work while some node are being compromised by the adversary.

Cryptography is often used to protect and secure the communications in the sensor network. We normally rely on keys to achieve this goal, so how to manage the keys in the networks is a very challenge problem. Since the wireless sensor node are limited in battery power, computing ability, and memory storage, it is reasonable to use the symmetric key cryptography as the fundamental tool to manage the keys.

One important issue we must address in the key management protocol that uses the symmetric key cryptography is how to establish a shared secret key between two end- nodes in the first place. This issue is also called the key agreement problem. We consider that the key predistribution scheme, in which the keys are preloaded in to the sensor nodes before their deployment, is the most practical approach to deal with the key agreement problem.

Base on the degree of keys sharing between the nodes in the networks, the key predistribution scheme based on the symmetric key cryptography can behave in quite different manners. Considered the situation that one global key shared by all the nodes in the network, and any pair of nodes can use the shared global key to encrypt or authenticate the data. This approach will have the lowest storage cost and be very energy-efficient since no communication is needed for the exchange of keying materials, while the radio communication is considered as more energy consuming than the computation by several orders of magnitude [1]. However, it is obvious to see the insecure of the protocol since compromising one node will leak out the global key.

In the other hand, by preload each node with N-1 secret pairwise keys, each key is only known by one of the other N-1 nodes (assuming N is the total amount of nodes). This approach is perfect in the security point of view since the compromised nodes won't reveal any keying information that other pair of nodes used. However, this approach is not practical in the large network since the nodes only have limited memory. Furthermore, in the real world application, nodes will be added and deleted; these actions become very difficult because the existing nodes do not have the pairwise key with the new added nodes.

The above issue is a trade-off between efficiency and security, it is reasonable to believe that by preload each node the secret key with its immediate neighbor only, the scheme will have a reasonable efficiency and security level. Knowing which nodes are the immediate neighbors becomes important since the primary goal in the sensor network is to build up the pairwise key among neighbor nodes, so the secure communication can be preformed.

In this paper we proposed a key management scheme that uses in the static sensor networks. By incorporate the node deployment information into LEAP+ [2] to create a more resilient scheme against node cloning attack and HELLO flood attack, while the overall performance and the resource consumption are remain in the same level as LEAP+.

The rest of this paper is organized as follows: we discuss the relative works in Section 2, then gives a quick present on LEAP+ and the network models in Section 3. We present our scheme in Section 4, and give an analysis on security and performance in Section 4 and 5, and concluding this paper in Section 7.

2. RELATED WORK

There are numbers of key predistribution schemes proposed recently for the WSNs [2], [3], [4], [5]. In that, Eschenauer and Gligor [3] proposed a probabilistic key predistribution scheme that is simple, and provide an effective tradeoff between robustness and scalability. The idea is to choose a key ring of n keys from a large pool of keys, then assigned to each node, for any tow node to communicate they need to find a common key from their key rings, if there is no key in common, intermediate nodes involved, the drawback includes: load of the authentication process and doesn't define the process for revoking of refreshing keys. In 2003, Du *et al*. [4] proposed a key management scheme, it use the same paradigm as Eschenauer and Gligor, but instead of individual keys, it uses the concept of Blom's [6] key matrix. The benefit of it is that it more robust then Eschenauer and Gligor's scheme at a reasonable scalability cost. The disadvantage is its complexity, it is likely to use more energy due to its computational cost and on-demand key computation.

Zhu, Setia, and Jajordia introduced the localized encryption and authentication protocol (LEAP+) [2]. It use four types of keys to accomplish network-wide, group, and pairwise keying capabilities, it uses an preloaded master secret key to establish the pairwise keys with its immediate neighbors, and the master secret key will be delete after the node completes its neighbor discovering. Base on the assumption that the time for the adversary to compromise a node is longer than the time for a node's neighbor discovering, this scheme restricts the impact of node compromising attack to only the immediate neighbors nodes of the compromised node while keeps the energy and memory cost to an economic level. It also use μ TESLA [7] and one-way key chain for broadcast authentication, as well as key revocation and refresh, which makes it satisfy the requirements of WSNs. And the computation and storage cost is quite reasonable. Our scheme is based on LEAP+, an overview of it is given in Section 3.

The Scalable, Hierarchical, Efficient, Location-aware, and Light-weight (SHELL) protocol [8] is a complicated cluster-based key management scheme. It is influenced by LEAP with its use of multiple types of keys, and using the concept of node clustering. Each cluster has its own distributed key management entity in a non-cluster-head node. Thus, the operational responsibility are separated, leading to a better resiliency against node capture, but the price for key setup and communication processes are too complex to be described.

Location-based predistribution (LPD) schemes [5], [9], use location information to localize the impact due to node compromise attack. It also has higher connectivity than the uniform predistribution schemes because the connectivity in LPD is mainly decide by the deployment knowledge. The storage coast depends on the density of nodes in one cell. Liu and Ning provide two LPD solutions: closet pairwise keys scheme (CPKS), and location-based key predistribution (LBKP) in [5], CPKS assumes the deployment point of each node is known in advance, which is very hard for implementation. LKBP partitions a deployment area in multiple square areas, using the same key predistribution scheme proposed by Blundo [10].

Node fabricating and node cloning attack is destructive to the WSNs, since most scheme will let the node keeps a secret that shares with other nodes in the networks, makes it fragile to such attacks. Detection schemes have been proposed to install in the networks, but its accuracy is yet to be proved.

Our design goal is mainly focus on fixing the security problem that LEAP+ could possibly suffered, and provide a solution that is as efficient as LEAP in the large distributed sensor networks.

3. SYSTEM DESCRIPTION

We first give an overview of LEAP+ in Section 3.1, a discussion on the security for LEAP+ in Section 3.2. The nodes in the sensor networks are all stationary.

3.1 LEAP+

In the sensor network, the package delivered in it can be classified in to several categories by its purpose. Different kind of package will require different kind of security requirements, for example, the readings reported by nodes to the base station normally require data confidentiality while the routing information usually doesn't need the data confidentiality. In order to provide all of the security requirements those are needed in the sensor network, LEAP+ supports the establishment of four kinds of keys for each sensor nodes: an individual key that share with the base station, a pairwise key shared with particular neighbors, a cluster key shared with every immediate neighbor, and a global key that is shared for all the node in the networks.

We focus on establishing the pairwise keys in LEAP+, it has three assumptions: First, the nodes in the networks are stationary, which means that the neighbors of a node is relatively static, so a new added sensor node will find most of its neighbors at the time it's been initially deployed. Second, the base station cannot be compromised, which is critical since our scheme rely much on the predistributed secrets. Third, the nodes that deployed in a hostile environment must be manufactured to sustain the node compromise attack for at least a few seconds; or the adversary can easily compromise most of the nodes in the network and taking control over the network. We assume that there exists a time interval T_{min} that adversary must take the time more than T_{min} to compromise the node, and the time T_{est} for the newly added node to find its immediate neighbor is smaller than T_{min}, which means $T_{min} > T_{est}$.

LEAP+ uses a pseudo random function (PRF) to compute most of the keys. We define a PRF as in [11]: it is a deterministic function f: $\{0, 1\}^n \rightarrow \{0, 1\}^n$ which is efficient (i.e. computable in polynomial time) and takes two inputs x, k are random n-bit stream. f cannot be distinguished from *random* functions by any probabilistic polynomial-time algorithm that asks and receives the value of a function at arguments for its choice.

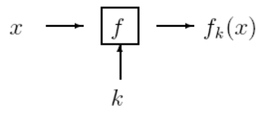

Figure 1. Pseudorandom function.

The basic pairwise key establishment consists of four steps:

Key Predistribution. The base station generates an initial key K_{IN} and loads it into the new node u, the node will calculate its master key K_u by using it is node id $K_u=F_{K_{IN}}(u)$.

Neighbor Discovery. Right after *u* is being deployed, it tries to discovery its neighbors by broadcasting a HELLO message that contains its id. It also starts a timer that will fire after T_{min}. and trigger key erasure phase. Node *u* waits for each neighbor *v* to respond to the HELLO message with an ACK message that includes its id, v. The ACK of *v* is authenticate by its master key K_v, since u knows $K_{IN,}$ it can also derive K_v and verify *v*'s identity.

$$u \longrightarrow * : \quad u.$$
$$v \longrightarrow u : \quad v, MAC(K_v, u|v).$$

Pairwise Key Establishment. Node u compute its pairwise key with *v* as $K_{uv}= F_{Kv}(u)$. Node *v* can compute K_{uv} in the same way.

Key Erasure. When the timer fires after T_{min}, node u erases K_{IN} and neighbors' master keys, and keeps its own master key. Note that the paper said that node u does not have to authenticate itself to node v. because any further messages from node u authenticated with K_{uv} will prove node *u*'s identity. After these phases the establishment of pairwise key between u and *v* is done.

3.2 Security Analysis

Unlike the random predistribution scheme we have discussed, LEAP+ will allow every legal node u (nodes have K_{IN}) to establish a pairwise key with each of its neighbors, after *u* erase the K_{IN}, compromise *u* will only get the pairwise keys for node *u*'s neighbors, and no additional pairwise key can the compromised node establish. Moreover, the compromise of one node will not leak out any information about other pariwise keys in the network.[1]

[1] An implicit assumption here is that a sensor node is able to erase the key completely. Another implicit assumption is that node *u* will not keep the master key of another node *v*.

It is obvious that LEAP+ will suffer from HELLO flood attacks, although LEAP+ has proposed to use the global key that shared by every node in the network. Adversary only need to compromise one node to launch this attack. Another possible threat is the replay attack. Because node *u* doesn't authenticate itself to *v* immediately, adversary can copy the message that is encrypted by K_{uv}, and use the id of node *u* to broad an HELLO message near v, and send back the message encrypted by K_{uv} back to node *v*, node *v* will accept another *u* as its legal neighbor.

Base on the above analysis, we first introduce an authenticating step to eliminate the possibility to use the replay attack. Furthermore, we minimize the area size that an adversary can launch the HELLO flood attacks by utilizing the deployment information.

4. PROTOCOL DESCRIPTION

Based on LEAP+ we describe above, we propose a new key predistribution scheme that utilizes the deployment information. We first describe the network model and the adversary model in Section 4.1 and 4.2, and then give our keying establishment scheme in Section 4.3.

4.1 Network Model

The entire network is divided into N_1 non-overlapping square cells, each node in the network is identified by a coordinate (n_1, n_2), $0 < n_1 < N_1$, nodes with the same cell will have unique id n_2, we define n_2 as a possible integer. Note that the coordinate is predetermined before its deployment by the base station, and once it is deployed, the coordinate cannot be changed. Cell ids are assigned in a fixed order such that each cell id acts like a coordinate in a two-dimensional plane. Figure 2 illustrates an example of the network model.

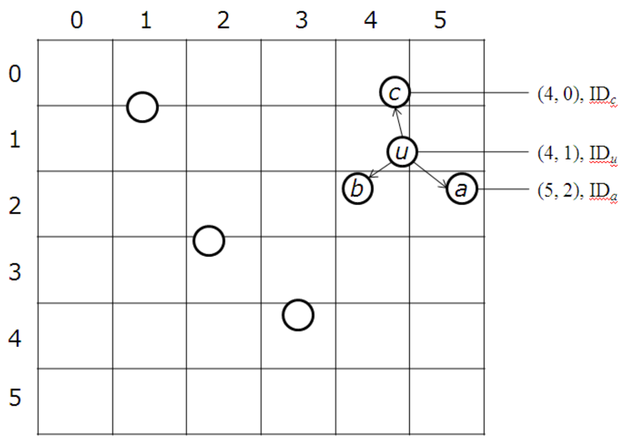

Figure 2. Network model of 6*6 grids.

4.2 Adversary Model

Due to the broadcast communication nature of WSNs, it is easy for adversary to eavesdrop any message that is transmitting between nodes. Moreover, even with tamper resistance device in installed in each node, it is still not a guarantee to perfect security of secrets [12], so node compromise is inevitable, but node itself should be able to sustain the compromise action for at least few second. Moreover, adversaries are able to clone the node that was just compromised, and insert the clone nodes into the networks, it is so called node cloning attack or node replicating attack. By combining the node cloning attack with HELLO flood attack, adversary can launch a powerful DoS attack.

4.3 Key Establishment

We first list the notations that we are going to use in the rest of the chapter:

Table 1. Notations.

Notation	**Description**
N	Number use only once.
u, *v*	Principles such as communicating node
f_k	Family of pseudo-random functions
$\{s\}_k$	Encrypt message s with symmetric key k
$MAC_k(s)$	Message authentication code of message s with symmetric key k
ID_u	The coordinate used by u: (u_1, u_2)

4.3.1 Establishing Individual Keys

Every node needs to keep an individual key that is only shared with the based station. This key is generated and preloaded into each node prior to its deployment. We can build up the individual key for each node as: $f_K(u_1||u_2)$. Here we refer K as a master key that only known to the base station. In our scheme, the base station can save the memory storage by not keeping a list of individual keys but only an id list, when it needs to communicate with the node, it can derive the individual keys on the fly.

4.3.2 Establishing Pairwise Keys

Our scheme consists of six steps:

Secret Predistribution: Before the deployment, the base station will determine the coordinate for each node *u*: $ID_u = (g_u, id_u)$, the coordinate includes the location of the deployment area g_u, and the node id id_u, which is unique in current grid, Base Station also generate an initial key K_{IN} and loads it into the node u, it will calculate its master key K_u by using it is node id $K_u = f_{K_{IN}}(ID_u)$.

Neighbor Discovery: After *u* is been deployed, *u* broadcasts an HELLO message to find out its immediate neighbor, at the same time *u* starts its timer that will fire after time $T_{min.}$ Neighbor *v* who receives the HELLO message will check the neighbor grid list to find out whether u is adjacent or not, if true then replies with the ACK message, ACK: N_1, $MAC(K_v, ID_u||ID_v)$. Notice that we use a nonce N_1 to prevent the replay attack, which we will describe in the node authentication phase. For the node that doesn't have K_{IN}, it cannot derive the correct K_v from id *v* to calculate the correct MAC value of $ID_u||ID_v$, and this false MAC will be detect by u, so the adversary cannot impersonate as a legal neighbor.

Pairwise Key Establishment: Node *u* computes the pairwise key K_{uv} with neighbor *v* as $K_{uv} = f_{K_v}(u_2)$. Node *v* can compute K_{uv} in the same way.

Key Erasure: When *u*'s timer fires after T_{min}, *u* erase the initial key K_{IN} and every master key K_v of its neighbors, but keeps its own master key K_u.

Node Authentication: *u* broadcast an authentication message to authenticate itself to the neighbors, with the specific location of u: AUTH: ID_u, $MAC(k_{uv}, N_1)$. If *u* is an illegal user (doesn't have K_{IN}), it cannot derive the correct K_{uv} to calculate the correct MAC value of N_1, and will be detect by *v*, so the adversary cannot impersonate as a new joining node.

Neighbor Grid List Refresh: Node *u* and *v* will launch a procedure to sum up the neighbor grid that appears during the neighbor discovery phase, the maximum number of grids one node can be adjacent with is three, so if there are four different grids appears in the neighbor grid list, the grid with least neighbor nodes should be discarded.

After these six phase, the identities of node *u* and *v* is been authenticated and the pairwise key is been established. There is the chance that both *u* and *v* are added to the network in the same time, for example, node *u* receives *v*'s REQ message before node *u* sends ACK message to node *v*'s REQ, in that case, node *u* will simply cancel its respond and calculate Kuv as their pairwise key. We use the following flow chart to express our scheme:

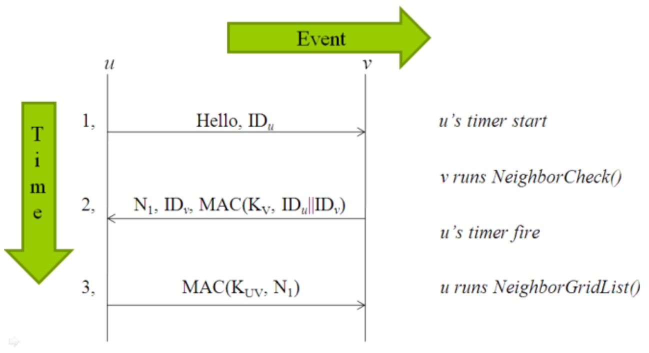

Figure 3. An illustration of protocol flow.

4.3.3 Establishing Cluster Keys

When the sensor nodes needs to broadcast the messages to the immediate neighbor such as the aggregated data or routing information data, pairwise key is not suitable to achieve the goal of one-to-may communication pattern. In order to protect the broadcasting message from being eavesdropped, each node needs to share a unique key with its immediate neighbors; this key is used to encrypt the broadcasting message. We call this key as the cluster key. To construct the cluster key is simple, after the node *u* completes the pairwise key establishment, *u* first generate a random key Kc^u, then encrypt it with the pairwise keys shared with each neighbor, and transmit it the encrypted key to neighbors v_1, v_2 …, v_m:

$$u \longrightarrow v_i: \{Kc^u\}_{K_{vi}}$$

Each node v_i decrypt the message, stores the key and send back their cluster keys to node u. this cluster key should be updated when the node revocation is invoked.

5. SECURITY ANALYSIS

In this section, we discuss the security property of our scheme to deal with three classes of attacks: replay attack, HELLO flood attack, Sybil attack, and node cloning attack. We first discuss the robustness of our key management protocol against various attacks in the network, and then study the survivability of the network when some nodes have been compromised and the clones have been made out.

5.1 Replay Attack

Due to the unattended nature of the sensor networks, it is easy for the adversary to capture any message that is broadcasting in the network. While the term replay attack usually uses in replaying the routing information to create loops and attract the traffic, here we discuss the replay on the keying message exchanged in the scheme.

In our scheme, every message exchanged during the key establishment phase except the HELLO message is different. Adversary catches the ACK message cannot replay it to other nodes' HELLO message since every node has an unique id. Adversary catches the AUTH message cannot authenticate itself to others because the nonce in AUTH will be different for each neighbor.

5.2 HELLO Flood Attack

Every node needs to broadcast a HELLO message to inform its neighbor about its presence and trying to create a pairwise key with each other. Neighbors receiving such a message will assume that it is within the radio range, calculate the necessary message and send back to the new joining node, such steps can be very memory and energy consuming. A laptop-class adversary can simply send HELLO message to a large radio range in high frequency to waste the CPU cycle, and fill up the queue in the node, furthermore, adversary can convince these nodes that he is their next hop neighbor, and misdirect the routing in the network.

Our scheme reduce the affected area by checking the grid number in the HELLO message is coming from the adjacent grid, if not the HELLO will be drop, there is a chance that the adversary is a laptop-class device and broadcast the HELLO message to a large zone that includes many grid, the neighbor grid list kept in each node will be able to verify the neighboring relation by runs the *NeighborGridList()* function to discard the location-impossible HELLO message.

5.3 Sybil Attack

Most key management schemes assume that nodes obtain one unique identity. In Sybil attack, adversary presents multiple identities and claim to be in multiple locations. Such an attack is effective in topology maintaining, and geographical routing since legal node can easily misdirect or confused by these fake identities and locations, this attack is base on the concept of *identity fraud*.

Our scheme is effective in preventing the unauthorized user (nodes without K_{IN}) to join the network, in the other hand, when unauthorized nodes receive the HELLO message from the new joining nodes, since K_v can only be calculate by correct id and the correct initialize key, which means that unless the adversary obtains the correct id-master key pair, he cannot deceive the legal joining nodes (joining nodes who poses K_{IN}) for a legal neighbor.

5.4 Node Cloning Attack

Adversary can replicate the secret he obtained from the compromised node and loads it into his own nodes or laptop, trying to establish a trust link with the legal nodes, or broadcasting error routing information, this is called node cloning attack or insider attack. This attack cannot be prevented nor detected since insider is considered a legal member in the network, so the best we can do is trying to reduce the number of the nodes that could be convinced by the insider. During the secret predistribution phase, we calculate the master key by the cell id and node id for node u:

$$K_u = F_{K_{in}}(ID_u), \text{ here } ID_u = (g_u, id_u)$$

We construct a binding-relation between cell id and node id in master key K_u, any further message that is encrypt by this master key will prove node u itself to the new joining node that node u is a legal user. Any new joining node possess the correct K_{IN} can verify this binding relation. According to the assumption that K_{IN} will no longer exist in the network after the new joining node has finish the neighbor discovery phase and delete K_{IN}, adversary cannot forge this relation by the wrong cell id- node id pair, thus the only thing he can do is to use the correct cell id- node id pair, and the comparable master key and put it into the cloning node, thus limit the effect area the adversary can reach consequently.

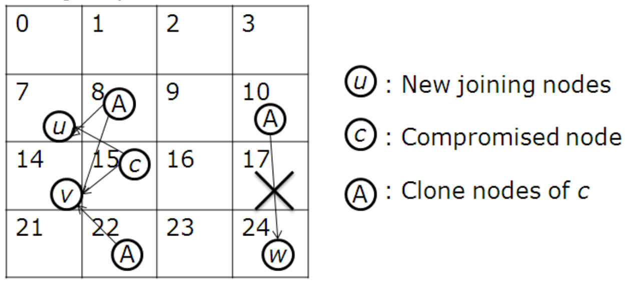

Figure 4. Links stats between the legal nodes and clone nodes at different grids.

6. SIMULATION

We first simulate the wireless sensor network scenario shown in figure 5 on NS2 platform to examine the survivability of LEAP+ and our scheme under HELLO flood attack

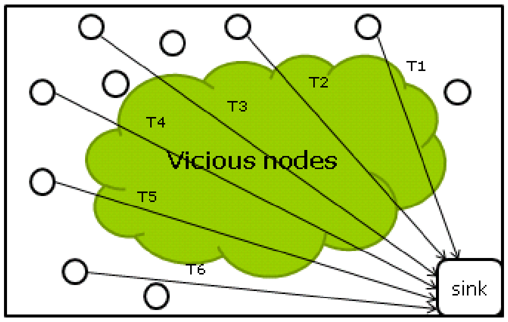

Figure 5. Network traffic and attack scenario.

There are 400 nodes uniformly distribute in an area of 500 meters * 500 meters, Network is divided into 5 * 5 square cells with the side length of 100 meters, the simulate time is 30 seconds with vicious nodes start at 5 second, the transmission frequency is 0.5

seconds, the result is shown in figure 6, while g represent the grid number we use in HELLO message.

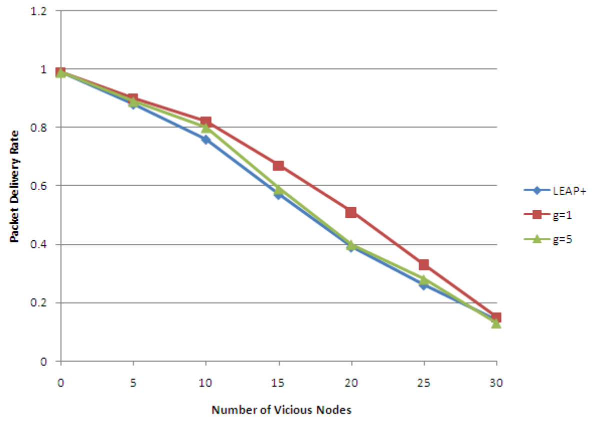

Figure 6. Packet delivery rate of different scheme.

Next we examine the fraction of new joining node that will be deceived by the node cloning attack with the following parameter: There are 200 new joining nodes in an area of 700 meters * 700 meters. and the network divided into 7 * 7 square cells with the size length of 100 meters the result is shown in figure 7

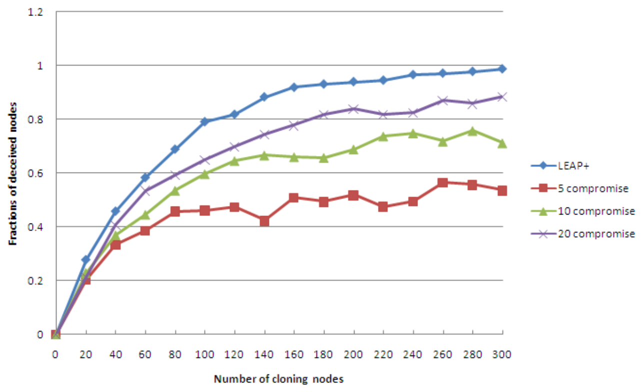

Figure 7. Fractions of deceived nodes of different scheme.

7. CONCLUSION

We have discussed the security concern of LEAP+ and present an improvement key establishment scheme for the static sensor network. The most benefit of our scheme is to build up a binding relation between the id and the location; this relation can be verify only by the legal user, the attacks due to identity fraud or compromised secrets can only work in the restricted area or fail to launch an attack at all. This property can increase the robustness of the sensor network, makes our scheme suitable to deploy in a hazardous area.

During the process of simulation we have found out the degradation of the packet delivery rate of the routing protocol will be affected by the amount of message exchanged during the key establishment due to severe packet collision, further analysis on different combination of keying scheme and routing protocol is needed.

For the future work we want to address the problem of the key refreshment, especially the refreshment of the initialize key since our whole security depends on it, alone with the defense of possible chosen-plaintext-attack that could possibly reveal the initialize key. An extend version on the dynamic sensor network is also needed.

8. REFERENCES

[1] D. Estrin, M. Srivastava, and A. Sayeed, "Wireless Sensors Networks, " *MobiCOM Tutorial*, no. 5. http://nesl.ee.ucla.edu/tutorials/mobicom02. 2002.

[2] S. Zhu and S. Setia, "LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *ACM Transaction On Sensor Networks*, vol. 2, No. 9**,** pp. 500-528, November, 2006.

[3] L. Eschenauer and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 41-47, 2002.

[4] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks," *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 42-51, October, 2003.

[5] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pp. 72-82. 2003.

[6] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Proceedings of EUROCRYPT'84, Advances in Cryptology, Lecture Notes in Computer Science*, vol. 209, pp. 335–338, 1984.

[7] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "Spins: Security Protocols for Sensor Networks," *In Proceedings of ACM Wireless Networks*, vol. 8, pp. 521–534, September, 2002.

[8] M. F. Younis and M. Eltoweissy, "Location-Aware Combinatorial Key Management Scheme for Clustered Sensor Networks," *IEEE Transaction On Parallel and Distributed Systems*, vol. 17, issue: 8, pp. 865-882, August, 2006.

[9] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-Aware Key Management Scheme for Wireless Sensor Networks," *Proceedings of Second ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04)*, October, 2004.

[10] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences," *Proceedings of CRYPTO'92, Advances in Cryptology, Lecture Notes in Computer Science*, vol. 740, pp.471–486, 1993.

[11] O. Goldreich, S. Goldwasser, S. Micali, "How to Construct Random Functions," *Journal of the ACM*, vol.33, no.4, p.792-807, 1986.