ISSN:0975-9646

An Efficient Key Management Scheme for Heterogeneous Sensor Networks

S.Gandhi ,D.Indira
Department of Computer Science and Engineering
Gudlavalleru Engineering College
Gudlavalleru—521356

***Abstract*—Previous research on sensor network security mainlyconsiders homogeneous sensor networks, where all sensor nodes have the same capabilities. Research has shown that homogeneous ad hoc networks have poor performance and scalability.The many-to-one traffic pattern dominates in sensor networks,and hence a sensor may only communicate with a small portion of its neighbors. Key management is a fundamental security operation.Most existing key management schemes try to establish shared keys for all pairs of neighbor sensors, no matter whether these nodes communicate with each other or not, and this causes large overhead. In this paper, we adopt a Heterogeneous Sensor Network (HSN) model for better performance and security. We propose a novel routing-driven key management scheme, which only establishes shared keys for neighbor sensors that communicate with each other. We utilize Elliptic Curve Cryptography in the design of an efficient key management scheme for sensor nodes. The performance evaluation and security analysis show that our key management scheme can provide better security with significant reductions on communication overhead, storage space and energy consumption than other key management schemes.**

I. INTRODUCTION

WIRELESS sensor networks have applications in many areas, such as military, homeland security, health care, environment, agriculture, manufacturing, and so on. In the past several years, sensor networks have been a very active research area. Most previous research efforts consider homogeneous sensor networks, where all sensor nodes have the same capabilities. However, a homogeneous ad hoc network suffers from poor fundamental limits and performance. Research has demonstrated its performance bottleneck both theoretically [1], [2] and through simulation experiments and test bed measurements [3]. Several recent work (e.g., [4], [5],and [6]) studied Heterogeneous Sensor Networks (HSNs),where sensor nodes have different capabilities in terms of communication, computation, energy supply, storage space, reliability and other aspects .Security is critical to sensor networks deployed in hostile environments, such as military battlefield and security monitoring. A number of literatures have studied security issues in homogeneous sensor networks, e.g., [6], [7]. Key management is an essential cryptographic primitive upon which other security primitives are built. Due to resource constraints, achieving such key agreement in wireless sensor networks is non-trivial. In [6],For example, when $P$is 10,000, each sensor needs to pre-load more than 150 keys for a key-sharing probability of 0.9 [6]. If the key length is 256 bits, then 150 keys require a storage space of 4,800 bytes. Such a storage requirement is too large for many sensor nodes. In this paper, we present an efficient key management scheme that only needs small storage space. The scheme achieves significant storage saving by utilizing 1) the fact that most sensor nodes only communicate with a small portion of their neighbors; 2)an efficient public-key cryptography. Below we briefly discuss the two issues. More details are given in Sections II and III. Most existing sensor key management schemes are designed to set up shared keys for all pairs of neighbor sensors, without considering the actual communication pattern. In many sensor networks, sensor nodes are densely deployed in the field.

Definition 1. c- neighbor:
A neighbor sensor node v is referred to as a communication neighbor (c-neighbor) of sensor node u if v is in a route from u to the sink. Based on the above observation, we propose a novel idea for efficient key management in sensor networks. A key management scheme only needs to set up shared keys for each sensor and its c-neighbors, i.e., it does not need to set up shared keys for each pair of neighbor sensors. The recent implementation of 160-bit ECC on Atmel ATmega128, a CPU of 8Hz and 8 bits, shows that an ECC point multiplication takes less than one second [11], which demonstrates that the ECC public-key cryptography is feasible for sensor networks. Compared with symmetric key cryptography, public-key cryptography provides a more flexible and simple interface, requiring no key pre-distribution, no pair-wise key sharing, and no complicated one-way keychain scheme.ECC can be combined with Diffie -Hellman approach to provide key exchange scheme for two communication parties.ECC can also be utilized for generating digital signature, data encryption and decryption. The Elliptic Curve Digital Signature Algorithm (ECDSA) utilizes ECC to generate digital signature for authentication and other security purposes [12], [13]. Several approaches for encryption and decryption using ECC have been proposed [10], [12]. Please refer to references[10], [12], [13] for the details.In this paper, we present an efficient key management scheme for HSNs. The scheme utilizes the c-neighbor concept and ECC public-key cryptography. Typical sensor nodes are unreliable devices and may fail overtime. Our key management scheme considers topology change caused by node failures. That is, the scheme set up pair wise keys for each sensor with more than one neighbor. The contributions of this paper are three folds. First, we observed the fact that a sensor only communicates with a small portion of its neighbors and utilized it to reduce the overhead of key

management. Second, we designed an effective key management scheme for HSNs by taking advantage of powerful H-sensors. Third, we utilized a public key algorithm - ECC for efficient key establishment among sensor nodes. The rest of the paper is organized as follows.

Section II describes the routing structure in HSNs. Section III presents the routing-driving key management scheme. Section IV gives the simulation results and security analysis.

II. THE ROUTING STRUCTURE IN HSNS

In this Section, we present an efficient key management scheme for HSNs which utilizes the special communication pattern in sensor networks and ECC. The scheme is referred to

as ECC-based key management scheme. We consider an HSN consisting of two types of sensors: a small number of high-end sensors (H-sensors) and a large number of low-end sensors(L-sensors). Both H-sensors and L-sensors are powered by batteries and have limited energy supply. Clusters are formed in an HSN. For an HSN, it is natural to let powerful H-sensors serve as cluster heads and form clusters around them. First, we list the assumptions of HSNs below.

1) Due to cost constraints, L-sensors are NOT equipped with tamper-resistant hardware. Assume that if an adversary compromises an L-sensor, she can extract all key material, data, and code stored on that node.

2) H-sensors are equipped with tamper-resistant hardware. It is reasonable to assume that powerful H-sensors are equipped with the technology. In addition, the number of H-sensors in an HSN is small (e.g., 20 H-sensors and 1,000 L-sensors in an HSN). Hence, the total cost

of tamper-resistant hardware in an HSN is low.

3) Each L-sensor (and H-sensor) is static and aware of its own location. Sensor nodes can use a secure location service such as [14] to estimate their locations, and no GPS receiver is required at each node.

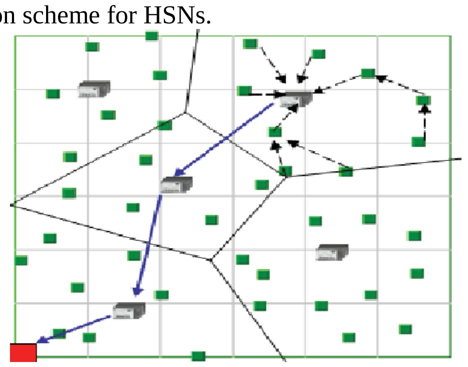
4) Each L-sensor (and H-sensor) has a unique node ID.

5) The sink is trusted.

The notations used in the rest of the paper are listed below.

1) u and v are L-sensors.

2) H is an H-sensor. Next, we briefly describe a cluster formation scheme for HSNs.



A. The Cluster Formation

After sensor deployment, clusters are formed in an HSN. We have designed an efficient clustering scheme for HSNs in [15]. Because of the page limit, we will not describe the details of the clustering scheme here. For the simplicity of discussion, assume that each H-sensor can communicate directly with its neighbor H-sensors All H-sensors form a backbone in an HSN. After cluster formation, an HSN is divided into multiple clusters, where H sensors serve as the cluster heads. An illustration of the cluster formation is shown in Fig. 1, where the small squares are L sensors, large rectangular nodes are H-sensors, and the large square at the bottom-left corner is the sink.

B. Routing in HSNs

In an HSN, the sink, H-sensors and L-sensors form a hierarchical network architecture. Clusters are formed in the network and H-sensors serve as cluster heads. All H-sensors form a communication backbone in the network. Powerful H-sensors have sufficient energy supply, long transmission range, high date rate, and thus provide many advantages for designing more efficient routing protocols. We have designed an efficient routing protocol for HSNs in [16]. Routing in an HSN consists of two phases: 1) Intra-cluster routing – each L-sensor sends data to its cluster head via multi-hops of other L-sensors; and 2) Inter-cluster routing - a cluster head (an H-sensor) aggregates data from multiple L-sensors and then sends the data to the sink via the H-sensor backbone. The routing structure in an HSN is illustrated in Fig. 1. We are interested in key establishment for L-sensors, so we briefly describe the intra-cluster routing scheme below. An intra-cluster routing scheme determines how to route packets from an L-sensor to its cluster head. The basic idea is to let all L-sensors (in a cluster) form a tree rooted at the cluster head H. It has been shown in [17] that:

(1) If complete data fusion is conducted at intermediate nodes,(i.e., two k bit packets come in, and one k-bit packet goes out after data fusion) then a minimum spanning tree (MST) consumes the least total energy in the cluster.

(2) If there is no data fusion within the cluster, then a shortest-path tree (SPT) consumes the least total energy.

(3) For partial fusion, it is a NP complete problem of finding the tree that consumes the least total energy.

III. THE ROUTING-DRIVEN KEY MANAGEMENT SCHEME

Key setup for L-sensors can be achieved in either centralized or distributed way. First, we present the centralized scheme.

A. Centralized Key Establishment

We propose the following centralized ECC-based key management scheme. A server is used to generate pairs of ECC public and private keys, one pair for each L-sensor (and H sensor).

The server selects an elliptic curve E over a large finite field F and a point P on that curve. Each L-sensor (denoted as u) is pre-loaded with its private key (denoted as $KR_u = I_u$).

An H-sensor has large storage space and is pre-loaded with public keys of all L-sensors (such as $KU_u = I_u P$). Each H sensor (denoted as H) also stores the association between every L-sensor and its private key. An alternative approach is to pre-load each L-sensor its public key and then let every L sensor sends the public key to H after deployment. However, this scheme introduces large communication overhead and furthermore security problems, since an adversary may modify the public key during its route to H.

Given the protection from tamper-resistant hardware, the same ECC Public / private key pair can be used by all H-sensors, which reduces the storage overhead of the key management scheme. Each H-sensor is pre-loaded with a pair of common ECC public key (denoted as $K_{UH} = I_H P$) and private key (denoted as $K_{RH} = I_H$). The public key of H-sensors – K_{UH} is also loaded in each L-sensor, and the key is used to authenticate broadcasts from H-sensors. The ECDSA algorithm [13] isused for authenticating broadcasts from H-sensors. After selecting a cluster head H, each L-sensor u sends to H a clear (un-encrypted) *Key-request* message, which includes the L-sensor ID - u, and u' s location. Next, H generates shared-keys for each L-sensor and its c-neighbors. For an L sensor u and its c-neighbor v, H generates a new key $K_{u{\cdot}v}$.

Recall that H is pre-loaded with the public keys of all L sensors. H encrypts $K_{u{\cdot}v}$ by using u's public key and an ECC encryption scheme [18], and then H unicasts the message to u. L-sensor u decrypts the message and obtain the shared key Between itself and v. After all L-sensors obtain the shared keys, t hey can communicate securely with their c-neighbors.

B. Distributed Key Establishment

The key setup can also be done in a distributed way. In the distributed key establishment, each L-sensor is pre-loaded with a pair of ECC keys - a private key and a public key. When an L-sensor (denoted as u) sends its locations information to its cluster head H, u computes a Message Authentication Code (MAC) over the message by using u's private key, and the MAC is appended to message. When H receives the message, H can verify the MAC and then authenticate u's identify, by using u's public key. Then H generates a certificate (denotedas CAu) for u's public key by using H's private key. After determining the routing tree structure in a cluster, the cluster head H disseminates the tree structure (i.e., parent child relationship) and the corresponding public key certificate to each L-sensor. The public key certificates are signed by H's private key, and can be verified by every L-sensor, since each L-sensor is preloaded with H's public key. A public key certificate proves the authenticity of a public key and further proves the identity of one L-sensor to another L-sensor. If two L-sensors are parent and child in the routing tree, then they are c-eighbors of each other, and they will set up a shared key by themselves. For each pair of c-neighbors, the sensor with smaller node ID initiates the key establishment process. For example, suppose that L-sensor u and v are c neighbors and u has a smaller ID than v. The process is presented below:

1) Node u sends its public key K_U
$_u = I_u P$ to v.
2) Node v sends its public key K_U
$_v = I_v P$ to u.
3) Node u generates the shared key by multiplying its private key I_u with v's public key – $K_{U\,v}$, i.e., $K_{u{\cdot}v} =$
$K_{Ru}\, K_U$
$_v = I_u I_v P$; similarly, v generates the shared key
- $K_{u{\cdot}v} = K_R$
$_v\, K_U$
$_u = I_u I_v P$.

After the above process, nodes u and v share a common key and they can start secure communications. To reduce the computation overhead, symmetric encryption algorithms are used among L-sensors. Note that in the distributed key establishment scheme, the assumption of having tamper-resistant hardware in H-sensors can be removed.

C. Key Revocation

When an L-sensor is compromised by an adversary, all the keys used by this L-sensor needs to be revoked. Assume that the node compromise is detected by some scheme and is reported to the cluster head H. A digital signature (denoted as *sign*) is calculated over the message by using the ECDSA algorithm [13] and H's private key , and the *sign* is appended after the key list. The format of the *Revocation* message is: Node ID + *sign*. Upon receiving a *Revocation* message, an L-sensor checks whether it communicates with the compromised node. If so, the L sensor revokes the keys shared between them. *Revocation* message, it can check the integrity of the message by verifying the digital signature. This prevents an adversary from sending a fake *Revocation* message.

IV. Performance Evaluation

In this Section, we present the performance evaluation results of the ECC-based key management scheme (referred to as the ECC scheme below). The key pre-distribution scheme proposed by E schenauer and Gligor [6] is used for comparison, and it is referred to as the E-G scheme. We compare the storage requirement and energy consumption in subsection A and B, respectively. The security analysis is presented in subsection C.

A. Significant Storage Saving

Assume that the number of H-sensors and L-sensors in an HSN is M and N, respectively. Typically we have $M << N$. In the centralized ECC key management scheme, each Lsensor
is pre-loaded with its private key and the public key of H-sensors. Each H-sensor is pre-loaded with public keys of all L-sensors, plus a pair of private/public key for itself, and a key K_H for newly deployed sensors. Thus, an H-sensor is pre-loaded with $N + 3$ keys. The total number of pre-loaded keys is:

$M \square (N + 3) + 2 \square N = (M + 2)N + 3M$ (1)

In the distributed ECC key management scheme, each Lsensor is pre-loaded with its public/private key. Each H-sensor is pre-loaded with public/private key and key K_H. Thus, the total number of pre-loaded keys is: $3M + 2N$ (2)

In the E-G scheme, each sensor is pre-loaded with m keys.The total number of pre-loaded keys in a network with $M+N$ sensors is:

$$m(M+N) \quad (3)$$

The value of m depends on the key pool size P and the probability of sharing at least one key between two sensors. When P is 10,000, m needs to be larger than 150 to achieve a key sharing probability of 0.9 [6]. Let's use an example to compare the storage requirement of ECC key management scheme and the E-G scheme. Suppose that there are $N=$ 1000 L-sensors and $M=$ 20 In a homogeneous sensor network with 1020 sensors, each sensor is pre-loaded with 150 keys. The total number of preloadedkeys is 153,000, which is 7 times more than that in the centralized ECC scheme, and about 74 times of that in the distributed ECC scheme. The example shows that the ECC key management scheme requires much less total storage space than the E-G scheme. In Fig. 2, we plot the total storage requirements for different sizes of sensor networks and different numbers of pre-loaded keys in the E-G scheme. The x-axis is m - the number of pre-loaded keys in the E-G scheme. The y-axis represents the total storage space required for pre-loaded keys (in the unit of key length). The top five dotted curves (with small circles) are the total required storage spaces under the E-G scheme, where $N=$ 1,000, 800, 600, 400, and 200 from top to bottom, respectively. The five solid lines at the bottom of Fig. 2 are thetotal required memory spaces under the centralized ECC key management scheme, for the five value of N (1,000, 800, 600,400, and 200). Fig. 2 shows that the ECC key anagement scheme requires much less storage space for pre-loaded keys than the E-G scheme, for different network sizes and numbers of pre-loaded keys (m) tested. The more keys pre-loaded in a sensor under the E-G scheme, the larger the storage saving achieved by the ECC scheme.

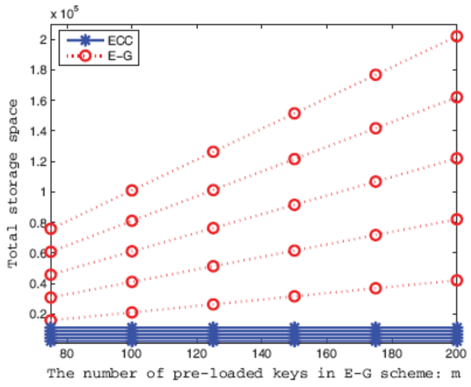
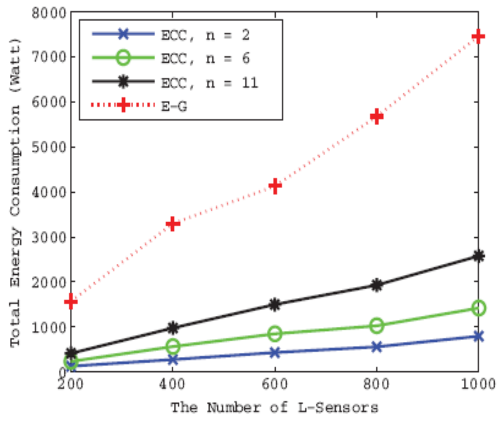


Fig. 2. Comparison of required storage space

B. Total Energy Consumption

We run simulations to compare the energy consumption of our ECC key management scheme and the E-G scheme. The simulations are conducted by using the QualNet simulator [19]. The default simulation testbed has 1 sink and 1000 L-sensors randomly distributed in a $1000m$ □ $1000m$ area. The underlying medium access control protocol is IEEE 802.11 Distributed Coordination Function (DCF). For the ECC scheme, there are additional 20 H-sensors. For comparison, 20 L-sensors are added for the E-G scheme. The transmission range of an L-sensor and an H-sensor is $60m$ and $150m$, respectively. The average number of neighbors for an L-sensor is 1000 □ □ □ 602<(1000□ 1000) □ 11. Each simulation run lasts for 600 seconds, and each result is averaged over ten random network topologies. The energy consumption parameters are set according to the MICA2 Mote datasheet [20]. The energy consumed to receive a packet is Erx =32mW, and the transmitter energy consumption is Etx =81mW. The idle power consumption is $Ps = 12mW$. We compare the total energy consumption of using the centralized ECC key management scheme and the E-G scheme. The energy consumption reported here only includes the energy used to set up security keys, but does not include the energy for data communications. In the simulation, the number of L-sensors varies from 200 to 1000, with an increase of 200. The number of H-sensors under the ECC scheme is always 20. For the E-G scheme, the key pool size is P= 10,000, and the number of pre-loaded keys in each sensor is $m = 150$, thus, the key-sharing probability is about 90%. Under the ECC scheme, a sensor only establishes shared key with communication neighbors. Denote the number of communication neighbors as n. We measure the energy consumption of the ECC scheme for different values of n, including 2, 6 and 11, where 11 means that a sensor sets up keys with every neighbor.



.

Fig. 3 shows that the ECC key management scheme consumes much less energy than the E-G scheme (including the case when n = 11), and the ECC scheme achieves more energy saving for larger networks. We obtain similar results for the distributed ECC key management scheme.

C. Security Analysis

In this subsection, we analyze the resilience of our ECC key management scheme against node compromise attack. We want to find out the effect of c L-sensors being compromised on the rest of the network. I.e., for any two L-sensors u and v which are not compromised, what is the probability that the adversary can decrypt the communications between u and v when c L-sensors are compromised? In the ECC key management scheme, each L-sensor is preloaded with one unique private key. After key setup, each pair of communicating L-sensors has a different shared key.Thus, compromising c L-sensors does not affect the security of communications among other L- sensors. In [7], Chan *etal▷* calculate the probability that two sensors have exactly j common keys in the E-G scheme is $p(j)$ =(pj)(p-j 2(m-j))(2(m-j) (m-j)/(p m)2 where $m \bigvee^{2}$, where m is the number of pre-loaded keys in each sensor. Chan *etal▷* give the probabilityof compromising a secure link under the E-G scheme as:

$C(m)$ =□(1-(1-m/p)c)j p(j)/□p(j)

In Fig. 4, we plot the probability that an adversary can decrypt the communications between two sensors u and v when c L-sensors (other than u and v) are compromised (referred to as compromising probability). In Fig. 4, the key pool size P is 10,000, and the number of compromised sensors - c varies from 10 to 200, with an increment of 10. For the E-G scheme, we calculate the probability for three different values of m: 20, 30, and 50. Fig. 4 shows that the more keys pre-loaded in a sensor under the E-G scheme, the larger the compromising probability, that is, less resilient to node compromise attack. For the ECC scheme, the compromising probability is always zero, no matter how many sensors are compromised, since each L-sensor uses a distinctive public/private key pair. Thus, the ECC key management scheme is very resilient against node compromise attack.

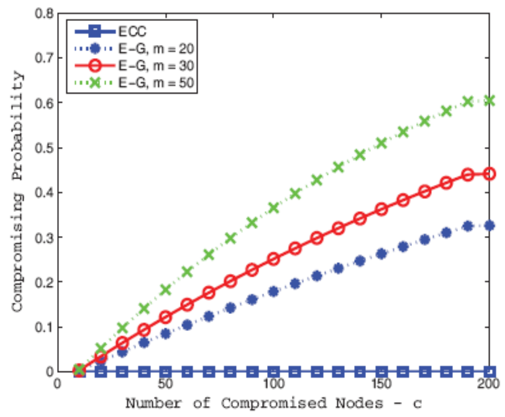


Fig. 4. The probability of an independent secure link being compromised.

V. CONCLUSIONS

In this paper, we presented an efficient key management scheme for heterogeneous sensor networks. The proposed key management scheme utilizes the fact that a sensor only communicates with a small portion of its neighbors and thus greatly reduces the communication and computation overheads of key setup. A public key algorithm – Elliptic Curve Cryptography (ECC) is used to further improve the key management scheme. The scheme only pre-loads a few keys on each L-sensor and thus significantly reduces sensor storage requirement. Our performance evaluation and security analysis showed that the routing-driven, ECC-based key management scheme can significantly reduce communication overhead, sensor storage requirement and energy consumption while achieving better security (e.g., stronger resilience against node compromise attack) than a popular key management scheme for sensor networks.

Hile achieving better security (e.g., stronger resilience against node compromise attack) than a popular key management scheme for sensor networks.

REFERENCES

[1] P. Gupta and P. R. Kumar, “The capacity of wireless networks,” *IEEE Trans. Inform. Theory,* vol. IT-46, no. 2, pp. 388-404, Mar. 2000.

[2] E. J. Duarte-Melo and M. Liu, “Data-gathering wireless sensor networks: organization and capacity,” *Computer Networks (COMNET)* Special Issue on Wireless Sensor Networks, vol. 43, no. 4, pp. 519 537, Nov. 2003.

[3] K. Xu, X. Hong, and M. Gerla, “An ad hoc network with mobile backbones,” in *Proc. IEEE ICC 2002*, New York, NY, Apr. 2002.

[4] L. Girod, T. Stathopoulos, N. Ramanathan, *et al.*, “A system for simulation, emulation, and deployment of heterogeneous sensor networks,” in *Proc. ACM SenSys 2004*.

[5] M. Yarvis, N. Kushalnagar, H. Singh, *et al.*, “Exploiting heterogeneity in sensor networks,” in *Proc. IEEE INFOCOM 2005*, Miami, FL, Mar. 2005.

[6] L. Eschenauer and V. D. Gligor, “A key management scheme for distributed sensor networks,” in *Proc. 9th ACM Conference on Computerand Communication Security*, pp. 41-47, Nov. 2002.

[7] H. Chan, A. Perrig, and D. Song, “Random key predistribution schemes for sensor networks,” in *Proc. 2003 IEEE Symposium on Security and Privacy*, May 2003, pp. 197-213.

[8] J. M. Kahn, R. H. Katz, and K. S. J. Pister, “Mobile networking for smart dust,” in *Proc. ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom)*, Seattle, WA, August 1999, pp. 271-278.

[9] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler, “Hood: a neighborhood abstraction for sensor networks,” in *Proc. ACM International Conference on Mobile Systems, Applications, and Services (MobiSys ’04),* Boston, MA, June, 2004.

[10] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.