Transactions Papers

A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks

Xiaojiang Du, *Member, IEEE,* Mohsen Guizani, *Fellow, IEEE,* Yang Xiao, *Senior Member, IEEE,* and Hsiao-Hwa Chen, *Senior Member, IEEE*

***Abstract*—Previous research on sensor network security mainly considers 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.**

***Index Terms*—Security, key management, sensor networks, elliptic curve cryptography.**

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 testbed 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], Eschenauer and Gligor first present a key management scheme for sensor networks based on probabilistic key pre-distribution. Several other key pre-distribution schemes (e.g., [7]) have been proposed.

Probabilistic key pre-distribution is a promising scheme for key management in sensor networks. To ensure such a scheme works well, the probability that each sensor shares at least one key with a neighbor sensor (referred to as key-sharing probability) should be high. For the key pre-distribution scheme in [6], each sensor randomly selects its key ring from a key pool of size P. When the key pool size is large, each sensor needs to pre-load a large number of keys to achieve a high key-sharing probability. 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. For example, a smart dust sensor [8] has only 8K bytes of program memory and 512 bytes of data memory.

The above discussion shows that many existing key management schemes require a large storage space for key pre-distribution and are not suitable for small sensor nodes. In

Manuscript received August 17, 2006; revised February 16, 2007; accepted April 2, 2007. The associate editor coordinating the review of this paper and approving it for publication was S. Shen.

X. Du is with the Dept. of Computer Science, North Dakota State Univ., Fargo, ND 58105 USA (e-mail: dxj@ieee.org).

M. Guizani is with the Dept. of Computer Science, Western Michigan Univ., Kalamazoo, MI 49008 USA (e-mail: mguizani@ieee.org).

Y. Xiao is with the Dept. of Computer Science, Univ. of Alabama, Tuscaloosa, AL 35487 USA (e-mail: yangxiao@ieee.org).

H.-H. Chen is with the Department of Engineering Science, National Cheng Kung Univ., Taiwan (e-mail: hshwchen@ieee.org).

Digital Object Identifier 10.1109/TWC.2009.060598

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. One sensor could have as many as 30 or more neighbors [9]. The many-to-one traffic pattern dominates in most sensor networks, where all sensors send data to one sink. Due to the many-to-one traffic pattern, a sensor node may only communicate with a small portion of its neighbors, for example, neighbor sensors that are in the routes from itself to the sink. This means that a sensor node does not need shared keys with all neighbors. Below we give a definition that considers the fact.

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 new scheme can significantly reduce the overhead of key establishment in sensor networks. For example, suppose that a sensor node u has 30 neighbors but only sends packets to 2 neighbors (e.g., one primary next-hop node and one backup). Using traditional key management schemes, 30 pairwise of keys need to be established for u, one key for each neighbor. Using c-neighbor concept, only 2 pairwise keys need to be set up for u, one for each c-neighbor. Thus, the new scheme can significantly reduce communication and computation overheads, and hence reduce sensor energy consumption.

Public-key cryptography has been considered too expensive for small sensor nodes, because traditional public-key algorithms (such as RSA) require extensive computations and are not suitable for tiny sensors. However, the recent progress on Elliptic Curve Cryptography (ECC) [10] provides new opportunities to utilize public-key cryptography in sensor networks. 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 key chain 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 pairwise keys for each sensor with more than one neighbor. In case the primary next-hop node fails, a backup node is used for communications. In addition, if there is a need for two neighbor sensor nodes to set up shared keys later (e.g., in case all backup nodes fail); they can do this with the help from other neighbors [6].

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. Section V concludes this paper.

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.

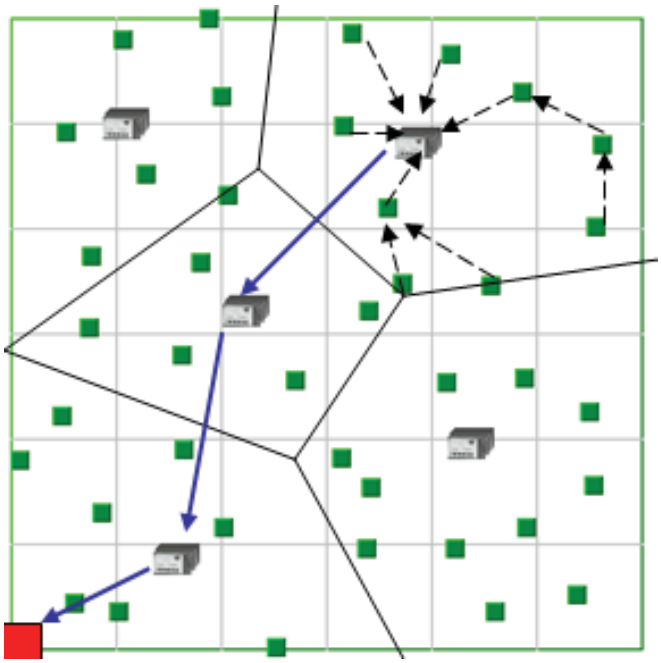

Fig. 1. Cluster formation in an HSN.

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 (if not, then relay via L-sensors can be used). 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.

For sensor networks where data generated by neighbor sensors are highly correlated (e.g., two k-bit packets are aggregated to one m-bit packet, where m is close to k), an MST can be used to approximate the least energy consumption case. To construct a MST, each L-sensor sends its location information to the cluster head H (during cluster formation phase), and then H can run a centralized MST algorithm to construct the tree. After constructing the MST, H disseminates the tree structure (parent-child relationships) to all L-sensors using one or more broadcasts. For example, a pair $(u,\ v)$ may be used to denote that L-sensor u is v's parent node. If the cluster is small, one broadcast message can include all the pairs. If the cluster is large, then it can be divided into several sections. H notifies L-sensors in each section by one broadcast. Note that the broadcast from a cluster head needs to be authenticated. Otherwise, an adversary may broadcast malicious messages and disrupt the dissemination of routing information. We discuss the broadcast authentication in next Section. For sensor networks where the data from neighbor sensors have little correlation, an SPT can be constructed, using either a centralized or distributed algorithm.

Since L-sensors are small, unreliable devices and may fail overtime, robust and self-healing routing protocols are critical to ensure reliable communications among L-sensors. During the tree setup, two or more parent nodes are determined for each L-sensor. One parent node serves as the primary parent, and other nodes serve as backup parents.

Given the tree-based routing structure within a cluster, each L-sensor only needs shared keys with its c-neighbors, i.e., its parent-nodes and child-nodes. In next subsection, we discuss how to establish shared keys.

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 $K_u^R = I_u$). An H-sensor has large storage space and is pre-loaded with public keys of all L-sensors (such as $K_u^U = I_u P$). 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.

The pre-loaded keys in H-sensors are protected by the tamper-resistant hardware. Even if an adversary captures an H-sensor, she could not obtain the key materials. 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 $K_H^U = I_H P$) and private key (denoted as $K_H^R = I_H$). The public key of H-sensors - K_H^U is also loaded in each L-sensor, and the key is used to authenticate broadcasts from H-sensors. The ECDSA algorithm [13] is used for authenticating broadcasts from H-sensors. When H broadcasts the routing structure information (e.g., the MST) to L-sensors, a digital signature is calculated over the message using H's private key. Each L-sensor can verify the digital signature by using H's public key, and thus authenticate the broadcast. In addition, each H-sensor is pre-loaded with a special key K_H, which is used by a symmetric encryption algorithm for verifying newly-deployed sensors and for secure communications among H-sensors.

After selecting a cluster head H, each L-sensor u sends to H a clear (un-encrypted) $Key\text{-}request$ message, which includes the L-sensor ID - u, and u's location. A greedy geographic routing protocol (e.g., [18]) may be used to forward the $Key\text{-}request$ message to H. Note that the location of the cluster head is known to all L-sensors during cluster formation. An L-sensor sends the message to the neighbor L-sensor that has the shortest distance to the cluster head, and the next node performs similar operation, until the packet arrives at the cluster head.

After a certain time, the cluster head H should receive $Key\text{-}request$ messages from all (or most) L-sensors in its cluster, and then H uses a centralized MST (or SPT) algorithm to determine the tree structure in the cluster. 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 $K_{u,v}$. Recall that H is pre-loaded with the public keys of all L-sensors. H encrypts $K_{u,v}$ 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, they 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 (denoted as 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-neighbors 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 $K_v^U = I_v P$ to u.
3) Node u generates the shared key by multiplying its private key I_u with v's public key - K_v^U, i.e., $K_{u,v} = K_u^R K_v^U = I_u I_v P$; similarly, v generates the shared key - $K_{u,v} = K_v^R K_u^U = I_u I_v P$.

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. H can disseminate a $Revocation$ message containing the identity of the compromised node. 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. Since each L-sensor knows H's public key, when an L-sensor receives the $Revocation$ 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 Eschenauer 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 L-sensor 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:

$$M \times (N + 3) + 2 \times N = (M + 2)N + 3M \quad (1)$$

In the distributed ECC key management scheme, each L-sensor 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:

$$3M + 2N \quad (2)$$

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 $M = 20$ H-sensors in an HSN. The total number of pre-loaded keys under the centralized and distributed ECC key management scheme is 21,060 and 2,060, respectively. In a homogeneous sensor network with 1020 sensors, each sensor is pre-loaded with 150 keys. The total number of pre-loaded keys 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.

We notice that a public-key cryptograph algorithm may need a longer key than a symmetric one to achieve similar security strength. For example, the security level of ECC with a 160-bit key is equivalent to that of a symmetric cryptograph algorithm using a 64-bit key. Even by considering the key length, the proposed ECC schemes still achieves a lot of storage savings compared to the E-G scheme. Using the parameters in the previous paragraph, the total storage space under the E-G scheme (with a 160-bit key) is about 2.9 times of that under the centralized ECC scheme (with a 64-bit key), and about 29.7 times of that under the distributed ECC scheme (with a 64-bit key).

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,

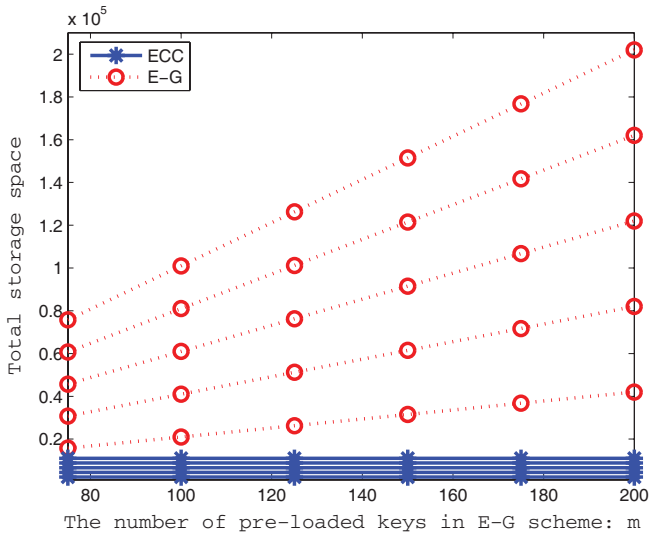

Fig. 2. Comparison of required storage space.

respectively. The five solid lines at the bottom of Fig. 2 are the total 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 management 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.

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 $1000m \times 1000m$ 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 $1000 \times \pi \times 60^2/(1000 \times 1000) \approx 11$. 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 E_{rx} =32mW, and the transmitter energy consumption is E_{tx} =81mW. The idle power consumption is $P_s = 12mW$.

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

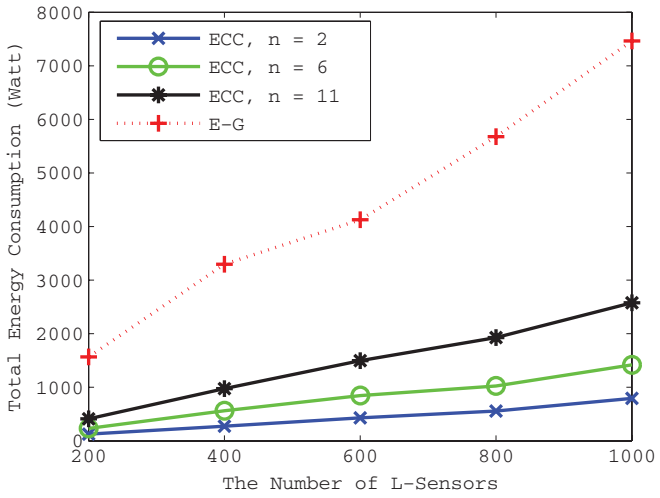

Fig. 3. Comparison of total energy consumptions.

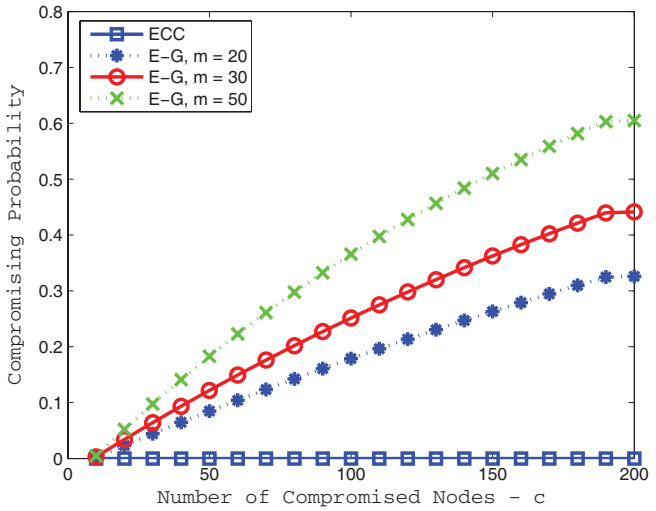

Fig. 4. The probability of an independent secure link being compromised.

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. The simulation results are reported in Fig. 3. 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 pre-loaded 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 $p(j) = \binom{P}{j}\binom{P-j}{2(m-j)}\binom{2(m-j)}{m-j} / \binom{P}{m}^2$, where m is the number of pre-loaded keys in each sensor. Chan $etal.$ give the probability of compromising a secure link under the E-G scheme as:

$$C(m) = \sum_{j=1}^{m}(1-(1-\frac{m}{\mathrm{P}})^c)^j p(j) \Big/ \sum_{j=1}^{m} p(j) \qquad (4)$$

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.

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.

VI. Acknowledgement

This research was supported in part by the US National Science Foundation (NSF) under grants CNS-0721907 and CNS-0709268, and the Army Research Office under grants W911NF-07-1-0250 and W911NF-08-1-0334.

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.
[11] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Proc. 6th International Workshop on Cryptographic Hardware and Embedded Systems*, Boston, MA, Aug. 2004.
[12] N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd ed. Graduate Texts in Mathematics, vol. 114, Springer, 1994.
[13] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society, Lecture Note Series 265, Cambridge University Press, 1999.
[14] L. Lazos and R. Poovendran, "SeRLoc: secure range-independent localization for wireless sensor networks," in *Proc. 2004 ACM workshop on Wireless security (ACM WiSe 2004)*, Philadelphia, PA.
[15] X. Du and F. Lin, "Maintaining differentiated coverage in heterogeneous sensor networks," *EURASIP J. Wireless Commun. and Networking*, no. 4, pp. 565-572, 2005.
[16] X. Du and Y. Xiao, "Energy efficient chessboard clustering and routing in heterogeneous sensor network," *International J. Wireless and Mobile Computing (IJWMC)*, vol. 1, no. 2, pp. 121-130, Jan. 2006.
[17] R. Cristescu and B. Beferull-Lozano, "Lossy network correlated data gathering with high-resolution coding," in *Proc. IEEE IPSN 2005*.
[18] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proc. 6th Annual International Conference on Mobile Computing and Networking*, pp. 243-254, 2000.
[19] QualNet simulator, Scalable Network Inc., www.qualnet.com
[20] MICA2 Mote Datasheet, www.xbow.com

Xiaojiang (James) Du is an Assistant Professor in the Department of Computer Science, North Dakota State University. Dr. Du received his B.E. degree from Tsinghua University, Beijing, China in 1996, and his M.S. and Ph.D. degrees from University of Maryland, College Park in 2002 and 2003, respectively, all in Electrical Engineering. His research interests are heterogeneous wireless sensor networks, security, wireless networks, computer networks, network and systems management, and controls. His research has been supported by the US National Science Foundation, Army Research Office and NASA. Dr. Du is an Associate Editor of WIRELESS COMMUNICATION AND MOBILE COMPUTING (Wiley), SECURITY AND COMMUNICATION NETWORKS (Wiley), JOURNAL OF COMPUTER SYSTEMS, NETWORKING, AND COMMUNICATIONS (Hindawi), and INTERNATIONAL JOURNAL OF SENSOR NETWORKS (InderScience). He is (was) the Chair of Computer and Network Security Symposium of IEEE/ACM International Wireless Communication and Mobile Computing Conference 2006 - 2009. He is (was) a TPC member for several major IEEE conferences such as INFOCOM, ICC, GLOBECOM, WCNC, IM, NOMS, BroadNet and IPCCC.

Mohsen Guizani is currently a full professor and chair of the Computer Science Department at Western Michigan University. He has authored or co-authored over 180 technical papers in major international journals and conferences. His research interests include computer networks, design and analysis of computer systems, wireless communications, and optical networking. He currently serves on the editorial boards of many national and international journals. He is the founder and Editor-In-Chief of Wiley WIRELESS COMMUNICATIONS AND MOBILE COMPUTING JOURNAL. He is a fellow of IEEE.

Yang Xiao (SM'04) is with Department of Computer Science, University of Alabama, USA. He is an IEEE Senior Member. Dr. Xiao was a voting member of IEEE 802.11 Working Group from 2001 to 2004. Currently, he serves as Editor-in-Chief for INTERNATIONAL JOURNAL OF SECURITY AND NETWORKS, INTERNATIONAL JOURNAL OF SENSOR NETWORKS, and INTERNATIONAL JOURNAL OF TELEMEDICINE AND APPLICATIONS. He is an Associate Editor or on editorial boards for five other journals. He served as a Guest Editor for eight journals' special issues. His research areas include wireless networks, mobile computing, network security, and telemedicine. He has published more than 190 journal/conference papers with more than 50 papers published in various IEEE journals/magazines.

Hsiao-Hwa Chen (S'89-M'91-SM'00) is currently a full Professor in the Department of Engineering Science, National Cheng Kung University, Taiwan. He obtained his BSc and MSc degrees from Zhejiang University, China, and a PhD degree from the University of Oulu, Finland, in 1982, 1985 and 1991, respectively. He has authored or co-authored over 250 technical papers in major international journals and conferences, five books and three book chapters in the areas of communications. He served as general chair, TPC chair and symposium chair for many international conferences. He served or is serving as an Editor or/and Guest Editor for numerous technical journals. He is the founding Editor-in-Chief of Wiley's Security and Communication Networks Journal (www.interscience.wiley.com/journal/security). He is the recipient of the best paper award in IEEE WCNC 2008.