EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Providing perfect forward secrecy for location-aware wireless sensor networks

Chi-Tung Chen[1], Shu-Yan Huang[2*] and Iuon-Chang Lin[2,3*]

Abstract

Sensor nodes are resource-constrained, such as low battery life, computation, bandwidth and memory, so traditional public key schemes are impractical in wireless sensor networks. In the previous schemes, symmetric cryptography is the most common method used in sensor nodes. How to distribute keys into every sensor node is an important issue in many applications for hierarchical sensor networks. Once adversaries compromise a sensor node, they can obtain all information from the sensor's memory, such as keying material. The revocation of compromised sensor nodes is also a necessary but troublesome operation. These compromised sensor nodes may lead to the compromise of the entire network. In this article, we present an efficient approach to establish security links between each sensor node/cluster head and its neighbor/member. Our scheme only requires small memory size for each cluster head and sensor node, and it can also ensure perfect forward secrecy via changing session key in every transmission.

Keywords: Wireless sensor network, Key management, Perfect forward secrecy.

Introduction

In recent years, wireless sensor network is an important issue in many applications, such as military intrusion detection, habitat monitoring, and so on. Sensor nodes are often deployed in unattended environments, so the security design is vital in many sensitive applications. The security mechanisms for wireless sensor networks have to provide authentication, confidentiality, integrity, scalability, and flexibility. Sensor nodes can sense and forward the readings to the base station (or sink), so the secure communication among sensor nodes is one of many important security issues in the sensor networks for the purpose of avoiding being eavesdropped or injected bogus data by adversaries. Many studies in the previous researches have been in the security issues, and key management has been a popular research so far [1-8].

Traditional asymmetric schemes such as public-key techniques are not suitable for the resource-constrained sensor nodes, which are characterized by limited memory, computation, communication, and power. There are many variations of symmetric key schemes [9-11] used in the certificate authentication, and verification of a broadcast message. These variations are suitable for sensor nodes because they use the delay disclosure key that is actually used in a symmetric scheme for authentication and verification.

The pairwise key establishment between any two neighboring nodes is the main objective. Each sensor node can communicate with each neighboring sensor node using the pairwise key they shared. Eschenauer and Gligor [7] proposed a well-known key management scheme called basic scheme. In the key predistribution phase, a large pool of P keys and their key identifiers are generated. Each sensor node randomly selects k keys from the key pool P without replacement. In the shared-key discovery phase, any two neighboring sensor nodes can find out if they share (at least) a common key via exchanging the list of key identifiers on their key rings or using a challenge-response protocol. If any two neighboring nodes can not find out a common key on their key rings, they can perform path-key establishment if the graph is connected. Chan et al. [1] proposed three schemes called q-composite random key predistribution, multipath key reinforcement and random-pairwise keys scheme, respectively. The first and second schemes are the modifications of the basic scheme [7]. The q-composite random key predistribution

*Correspondence: g9629001@mail.nchu.edu.tw; iclin@nchu.edu.tw
[2]Department of Management Information Systems, National Chung Hsing University, Taichung, Taiwan
[3]Department of Photonics and Communication Engineering, Aisa University, Taichung, Taiwan
Full list of author information is available at the end of the article

scheme requires that any two neighboring sensor nodes need to share at least q ($q > 1$) keys for their link in order to increase the resilience against sensor node compromise. The multipath key reinforcement scheme can strengthen any link between any two neighboring sensor nodes that shared a single key via updating the communication key if enough routing information of them can be exchanged. The random-pairwise keys scheme offers the perfect resistance against node capture and node-to-node authentication. These schemes are all based on probabilistic shared keys.

Perrig et al. [11] proposed two protocols called SNEP and μTESLA, respectively. SNEP uses a counter to achieve semantic security without transmitting the counter value. μTESLA employs a one-way key chain for the authentication of broadcast messages, and it is an important issue in wireless sensor networks. In [10], Liu and Ning proposed a variation of μTESLA called Multi-level μTESLA. This scheme improves the communication overhead, tolerance of message loss, scalability, resistance to replay attacks, and DOS attacks.

Heinzelman et al. [12] proposed a self-organizing clustering protocol called LEACH. This scheme can average energy consumption in homogenous wireless sensor networks. Each sensor node decides whether or not to become a cluster head during different cluster rounds. Hsieh et al. [9] proposed an adaptive security design based on LEACH, and they also used proposed intrusion detection module to detect the compromised cluster heads or sensor nodes by evaluating trust value. Oliveiraa et al. [13] proposed a scheme called SecLEACH to add security to LEACH. They used a random key predistribution scheme proposed in [7] to bootstrap security in LEACH.

Huan et al. [14,15] proposed the access control protocols in wireless sensor networks. They used ECC-based cryptography for sensor node authentication and pairwise key establishment. Any two neighboring sensor nodes can establish a pairwise key if each one is authentic. Zhu et al. [16] proposed a key management protocol called LEAP+ for sensor networks. They assumed that an adversary can not compromise a sensor node within a time interval $T_{\min}$. This scheme can also establish pairwise key between any two neighboring sensor nodes via exchanging their own identity. Suppose node x is a new deployed sensor node, and node y is a neighboring sensor node of node x, then they can establish pairwise key K_{xy} after neighbor discovery. If adversaries compromise node x, they do not have method to establish pairwise key with other sensor nodes by manipulating node x.

In ID-based cryptography [17], a user's ID is just like the user's public key. An ID-based signature scheme called BNN-IBS can be found in [18]. BNN-IBS is based on Schnorr signature [19], and this scheme can be efficiently used in wireless sensor networks without much computation overhead. Recently, Cao et al. [20] proposed a variation of BNN-IBS called vBNN-IBS with a smaller signature size. The schemes in [21,22] are the similar to ID-based cryptography.

In this article, we propose a secure communication scheme among nodes through preloading each node with a unique and private seed for a hierarchical (heterogeneous) sensor network. This scheme can achieve secure unicast, multicast, and local broadcast using the private seed which each sensor node possesses. When a cluster head is compromised by an adversary, we can redistribute the sensor nodes of this cluster into new cluster heads. Because each cluster head does not have the private seeds which its members posses, we can eliminate any compromised cluster head easily. Furthermore, our scheme can minimize the storage overhead of each sensor node by preloading each sensor node with one private seed only.

The rest of this article is organized as follows. In Section "Related works", we introduce related work. We present the background knowledge used in this article in Section "Preliminaries". In Section "The proposed method", we present our proposed method. Section "Security analysis" is the security analysis. Section "Performance evaluations" is the performance evaluation. The conclusion is in Section "Conclusion".

Related works

Du et al. [5] proposed a key management scheme for heterogeneous or hierarchical sensor networks. A large key pool and the corresponding key IDs are generated at the beginning. Each L-sensor is loaded with l keys, and each H-sensor (e.g., cluster head) is loaded with M ($M \gg l$) keys without replacement from the key pool. When the key predistribution phase is finished, the shared-key discovery phase is performed by each L-sensor and H-sensor for finding the pairwise key between any two nodes. In this article, we use the clustering method used in [5] to form clusters in the sensor networks. Du et al. [3,4] proposed a scalable and flexible pairwise key predistribution scheme. This scheme is more resilient against node capture than previous schemes.

In hierarchical sensor networks, exclusion basis system (EBS) applies a set of administrative keys to each sensor node [6]. The key management scheme is defined as EBS (n, k, m), where n is the number of the sensor nodes in the EBS, k is the number of administrative keys assigned to each sensor node, and m is the number of administrative keys not assigned to each sensor node. The total number of administrative keys is $k + m$. Each sensor node holds a unique subset of administrative keys. Chorzempa et al. [2] employed the EBS in their scheme, called SECK in hierarchical sensor networks. SECK is a cluster-based dynamic key management scheme. When one or more sensor nodes are compromised by adversaries, it has to rekey by AFN (i.e., cluster head). Once an AFN is lost or captured, each sensor node within the same cluster has to re-cluster, which is triggered by a trusted third party (TTP) (e.g., base station). SECK is resilient to sensor nodes and key captures. Our scheme is similar to SECK, each cluster is also controlled by the corresponding cluster head which stores some secret information, e.g., keys. Younis et al. [8] proposed a novel key management scheme called SHELL based on EBS [6] in clustered sensor networks. Command Node (e.g., sink or Base Station) designates for each cluster a number of key generating gateways (e.g., cluster head), so SHELL is more resilient against gateway compromise. They proposed a novel approach for administrative keys assignment in each cluster. The heuristic key assignment algorithm can efficiently resist the collusion attacks since each pair has the smallest Hamming distance between any two neighboring sensor nodes when assigning a subset of administrative keys to each sensor node.

In [21,23,24], the authors proposed several localization schemes. In this article, we assume that each sensor node can estimate its location by these localization schemes. We also assume that an adversary can not launch an efficient attack to affect the localization performance. In other words, each sensor node can estimate its location correctly.

Chang et al. [25] proposed a dynamic multicast communications scheme. In this article, we use this scheme for a secure multicast communication between any two neighboring sensor nodes. We introduce our network model and the scheme [25] in the next section.

Preliminaries

In this section, we briefly introduce our network architecture and the scheme in [25] called broadcast-encryption-based key management scheme as follows.

The network model

We present the hierarchical sensor network model in this section. The sensor network is composed of a base station, a small quantity of resource-rich cluster heads, and a large quantity of resource-constrained sensor nodes. Base station and cluster heads have more powerful energy, memory, and processing ability, but sensor nodes do not. The sensor nodes of a cluster gather information from the operational environment and send their readings to the cluster head. Then the cluster head collects the readings of these sensor nodes, and send them to the base station. We assume that even though all cluster heads are equipped with tamper-resistant hardware, they may be still compromised by adversaries. All sensor nodes are not equipped with tamper-resistant hardware because of the high cost. Cluster heads and sensor nodes are stationary, and sensor nodes may be distributed by airdropping or other methods. So we do not have any deployment knowledge about each sensor node. In other words, there is no way to know the neighbors of one sensor node in advance. We assume that each cluster head is reachable to all its members in its cluster, and each sensor node can communicate with its cluster head via one-hop or multi-hop transmission paths. The physical location of all sensor nodes and cluster heads are known [8]. In other words, all sensor nodes and cluster heads can be aware of their own location using the previous schemes such as [21,23,24]. Figure 1 shows that the hierarchical sensor network model used in this article.

The broadcast-encryption-based key management scheme

Chang et al. [25] proposed a broadcast scheme for secure multicast. We assume that the number of the broadcast group members is n, and U denotes the broadcast group, where $U = \{u_1, u_2, \ldots, u_n\}$. U_m denotes a multicast group, $U_m \in U$. For example, U_m can be $\{u_1, u_3\}$. An encryption algorithm denoted $E(*)$ is known to each one with a l-bits key. $E_K(M)$ denotes that a message M is encrypted with a l-bits key K. $H(*)$ is an one-way hash function with an output of a fixed length l-bits. $\|$ is a concatenation which can concatenate two or more strings together. We assume that the members $u_1, u_2, \ldots, u_n$ have the seeds $s_{u_1}, s_{u_2}, \ldots, s_{u_n}$ in advance. First, the sender selects a prime p_s arbitrarily from $p_1, p_2, \ldots, p_n$, a random number X, and a random secret key K. Second, the sender determines U_m and broadcasts $\{B, X, E_K(M)\}$, where $B = (p_s \prod_{u_i \in U_m} H(s_{u_i} \| X)) + K$. When a receiver u_x

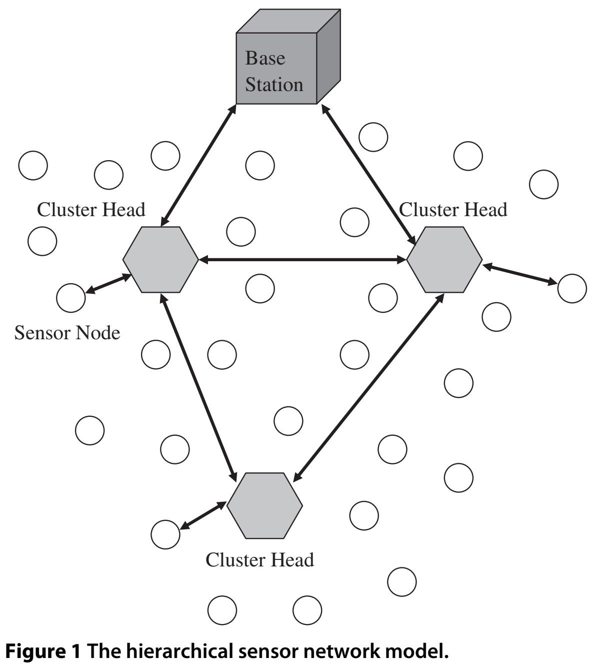

Figure 1 The hierarchical sensor network model.

receives $\{B, X, E_K(M)\}$, he can obtain K via computing the following equation:

$$K = B \bmod H(s_{u_i} \| X).$$

Note that the session key K should necessarily satisfy:

$$2^{l-1} < K < \min\{H(s_{u_i} \| X) \| u_i \in U_m\} \leq 2^l.$$

Finally, u_x is able to decrypt and get secret message M. Each sender can choose a random secret key K and random number X if they want to multicast secret messages in their group.

The adversary model and threat model

Adversaries are able to compromise (or capture) one or more sensor nodes (or cluster head) in wireless sensor networks. Then all the secret information (e.g., all key material, or data) held by the sensor node (or the cluster head) is known to the adversaries. Once adversaries obtain the secret keys from the compromised sensor nodes (or the cluster head), they may manipulate or attack the sensor network. We also assume that adversaries do not have any prior knowledge of what is stored in each sensor node [8]. In the previous scheme [2,5], once the adversaries compromise a cluster head, all the secret keys held by the cluster head in that cluster will be compromised. But our proposed scheme can prevent this situation from compromising all the keys in that cluster because each cluster head does not possess the private seeds held by its members. If an adversary compromises a cluster head, the sensor nodes in that cluster has to be re-clustered into new clusters and establish new security relationships among them. The adversary can also compromise a sensor node, and then the cluster head in that cluster has to revoke the compromised sensor node without the operation of rekeying. We assume that only the base station is trustworthy.

The proposed method

Our scheme

In this section, we describe our scheme designed for hierarchical sensor networks. Our scheme applies the location information to deploy the sensor nodes and cluster heads. The advantages of using location information are to prevent from replication attack, sybil attack, and wormhole attack. The detailed steps of our scheme are introduced in the following section.

The setup phase

Before sensor nodes and cluster heads are deployed, a TTP, e.g., the sensor networks controller or the base station, decides the system parameters such as a symmetric encryption algorithm $E(*)$ with a l-bit key, an one-way hash function $H(*)$ with a fixed l-bit output. We denote an ordinary sensor and a cluster head as N_i and CH_j, respectively. The base station preloads each sensor node N_i with two parameters including a unique identity ID_{N_i} and a seed S_{N_i} which is unique and private. For each cluster head CH_j, the base station also preloads it with a unique identity ID_{CH_j}, a unique and private seed S_{CH_j}, and another seed S'_{BS}, where $S'_{\mathrm{BS}} = H(S_{\mathrm{BS}} \| ID_{\mathrm{CH}_j})$; $\|$ is the operation of concatenation. Note that the base station has all private seeds of each cluster head and sensor node, i.e., S_{N_i}, S_{CH_j}, and its own private seed S_{BS} which is only known to itself.

The cluster head registration phase

After the setup phase is finished, all sensor nodes and cluster heads are uniformly and randomly deployed into a flat network that is a designated area. Cluster head CH_j has to inform the base station of its ID and location. The base station can authenticate the validity of each cluster head using the preloaded S'_{BS} in each cluster head. CH_j randomly chooses a prime RP_{CH_j}, a number X_{CH_j}, and a session key K_{CH_j}. Note that the session key K_{CH_j} has to be larger than RP_{CH_j} for the purpose of reducing the probability to illegally derive K_{CH_j}. Then CH_j sends the following message to the base station:

$$ID_{\mathrm{CH}_j}, \mathrm{Location}_{\mathrm{CH}_j}, B, X_{\mathrm{CH}_j}, E_{K_{\mathrm{CH}_j}}(ID_{\mathrm{CH}_j} \| \mathrm{Location}_{\mathrm{CH}_j}),$$

where $B = (RP_{\mathrm{CH}_j} \times H(S'_{\mathrm{BS}} \| X_{\mathrm{CH}_j})) + K_{\mathrm{CH}_j}$. After receiving the messages from each cluster head, the base station is able to compute $S'_{\mathrm{BS}} = H(S_{\mathrm{BS}} \| ID_{\mathrm{CH}_j})$, and then it can obtain K_{CH_j} via computing the following equation:

$$K_{\mathrm{CH}_j} = B \bmod H(S'_{\mathrm{BS}} \| X_{\mathrm{CH}_j}).$$

If S'_{BS} hold by cluster head CH_j is correct, the base station is able to decrypt $E_{K_{\mathrm{CH}_j}}(ID_{\mathrm{CH}_j} \| \mathrm{Location}_{\mathrm{CH}_j})$ with K_{CH_j}, and transmits the seeds of other cluster heads encrypted with the key K_{CH_j}, e.g., S'_{CH_m} (CH_m represents other cluster heads, $S'_{\mathrm{CH}_m} = H(S_{\mathrm{CH}_m} \| ID_{\mathrm{CH}_j})$, and $j \neq m$), to the cluster head CH_j for communications among cluster heads.

The clustering phase

Each cluster head broadcasts a *hello* message that contains ($ID_{\mathrm{CH}_j}, \mathrm{Location}_{\mathrm{CH}_j}$) to nearby sensor nodes using the maximum power with a random delay that can avoid the collision of *hello* messages [5], where $\mathrm{Location}_{\mathrm{CH}_j}$ represents the CH_j's location. If two or more cluster heads are available for sensor node N_i, it chooses the cluster head, denoted as ch_i, whose *hello* message has the strongest signal to become a member of the cluster controlled by ch_i. Note that we assume that a sufficient number of cluster heads are deployed, so most sensor nodes in the sensor network can receive the *hello* message(s) from at least one or more cluster heads. Finally, each cluster is controlled by a cluster head. This clustering scheme is similar to the schemes used in [5] or [9].

The sensor node join phase

After the clustering phase, each sensor node has to join the most appropriate cluster for itself. Suppose that sensor node N_i wants to join the cluster controlled by cluster head ch_i, it sends a *join* message:

$$(ID_{N_i}, \mathrm{Location}_{N_i}, TS_{N_i}, ID_{\mathrm{ch}_i}, H(S_{N_i} \| ID_{N_i} \| ID_{\mathrm{ch}_i} \| \mathrm{Location}_{N_i} \| TS_{N_i})),$$

where TS_{N_i} is the timestamp, to the cluster head ch_i for the purpose of becoming a member in this cluster. When the cluster head ch_i receives all join messages from its members which want to join its cluster, it has to send a *request* message to the base station for the purpose of obtaining the seeds of its members. First, ch_i randomly chooses a prime RP_{ch_i}, a number X_{ch_i}, and a session key K_{ch_i}. Note that these parameters RP_{ch_i}, X_{ch_i}, and K_{ch_i} can vary in every transmission. After cluster heads or sensor nodes use these parameters, they will erase these parameters from their own memory immediately. Second, ch_i sends the *request* message to the base station. This message is described as follows:

$$ID_{\mathrm{ch}_i}, ID_{\mathrm{BS}}, B, X_{\mathrm{ch}_i}, E_{K_{\mathrm{ch}_i}}(\ldots \| < ID_{N_i}, \mathrm{Location}_{N_i}, TS_{N_i}, H(S_{N_i} \| ID_{N_i} \| ID_{\mathrm{ch}_i} \| \mathrm{Location}_{N_i} \| TS_{N_i}) > \| \ldots),$$

where $B = (RP_{\mathrm{ch}_i} \times H(S'_{\mathrm{BS}} \| X_{\mathrm{ch}_i})) + K_{\mathrm{ch}_i}$. After receiving this message, the base station first computes $S'_{\mathrm{BS}} = H(S_{\mathrm{BS}} \| ID_{\mathrm{ch}_i})$, and then it can obtain K_{chi} via computing the following equation:

$$K_{\mathrm{ch}_i} = B \bmod H(S'_{\mathrm{BS}} \| X_{\mathrm{ch}_i}).$$

After obtaining K_{chi}, the base station is able to decrypt the message:

$$E_{K\mathrm{ch}_i}(\ldots \| < ID_{N_i}, \mathrm{Location}_{N_i}, TS_{N_i}, H(S_{N_i} \| ID_{N_i} \| ID_{\mathrm{ch}_i} \| \mathrm{Location}_{N_i} \| TS_{N_i}) > \| \ldots).$$

Because the base station knows all private seeds predeployed in each sensor node, it can check if these hash values are equal to the values it computes. If any computed hash value differs from the original one, the base station will reject the message. The base station then collects the IDs and locations of sensor nodes, and tabulates each sensor node's ID and location over every cluster. Then the base station searches the corresponding seed S_{N_i} if sensor node N_i is legitimate, and sends the following message to ch_i:

$$ID_{\mathrm{BS}}, ID_{\mathrm{ch}_i}, E_{K_{\mathrm{ch}_i}}(\ldots \| < ID_{N_i}, S'_{N_i} > \| \ldots),$$

where $S'_{N_i} = H(S_{N_i} \| ID_{\mathrm{ch}_i} \| \mathrm{Location}_{\mathrm{ch}_i})$. After receiving this message from the base station, ch_i can decrypt this message and obtain the seeds of its members, i.e. S'_{N_i}, in this cluster controlled by ch_i. Finally, ch_i can tabulate each sensor node's ID, location, and S'_{N_i} in this cluster. Note that S'_{N_i} is not the private seed S_{N_i} possessed by N_i.

The sensor node discovery phase

All sensor nodes are unaware of their neighboring sensor nodes until they have been deployed. We do not have any deployment knowledge. First, sensor node N_i tries to find its one-hop neighboring sensor nodes within its transmission range, so it broadcasts a *hello* message that contains its $(ID_{N_i}, \mathrm{Location}_{N_i}, ID_{\mathrm{ch}_i})$ to its one-hop neighboring sensor nodes. We recall that ch_i is the cluster head that N_i belongs to. If the cluster head ID of one of N_i's neighbors, say N_k, is the same as N_i's cluster head ID, N_k will send a *reply* message that includes its $(ID_{N_k}, \mathrm{Location}_{N_k}, ID_{\mathrm{ch}_i})$ to N_i. Note that N_k will includes cluster head ch_i if ch_i is within N_i's transmission range. Then, sensor node N_i can collect all *reply* messages from its one-hop neighboring sensor nodes whose cluster head ID is the same as its in this cluster controlled by ch_i. This step can be described as follows:

$$N_i \rightarrow * : \mathrm{hello}(ID_{N_i}, \mathrm{Location}_{N_i}, ID_{\mathrm{ch}_i}).$$
$$N_k \rightarrow N_i : \mathrm{reply}(ID_{N_k}, \mathrm{Location}_{N_k}, ID_{\mathrm{ch}_i}).$$

Upon receipt of this message, N_i checks if the locations of its neighbors are within its transmission range. If it is true, N_i sends a *request* message that contains $(ID_{N_1}, ID_{N_2}, \ldots, ID_{N_k})$ to ch_i for the purpose of obtaining the seeds of its one-hop neighboring sensor nodes, i.e., S''_{N_k}. This *request* message do not have to be encrypted. Each sensor node sends this message to its cluster head via one-hop or multi-hop transmission path through multiple sensor nodes. We recall that the *request* message may include cluster head ch_i if it is within sensor node N_i's transmission range. ch_i waits for each member in this cluster to send the *request* message. After collecting all *request* messages from each one of its members, ch_i has to unicast a message that includes the seeds of N_i's neighbors to N_i. ch_i randomly chooses a prime RP_{ch_i}, a number X_{ch_i}, and a session key K_{ch_i}. This message is described as follows:

$$ID_{\mathrm{ch}_i}, \mathrm{Location}_{\mathrm{ch}_i}, ID_{N_i}, B, X_{\mathrm{ch}_i}, E_{K_{\mathrm{ch}_i}}(< ID_{N_1}, S''_{N_1} > \| < ID_{N_2}, S''_{N_2} > \| \ldots, < ID_{N_k}, S''_{N_k} >),$$

where $B = (RP_{\mathrm{ch}_i} \times H(S'_{N_i} \| X_{\mathrm{ch}_i})) + K_{\mathrm{ch}_i}$, and $S''_{N_k} = H(S'_{N_k} \| ID_{N_i} \| \mathrm{Location}_{N_i})$. After receiving this message, N_i

first computes $S'_{N_i} = H(S_{N_i} \| ID_{\text{ch}_i} \| \text{Location}_{\text{ch}_i})$, and then it can obtain K_{ch_i} via computing the following equation:

$$K_{\text{ch}_i} = B \bmod H(S'_{N_i} \| X_{\text{ch}_i}).$$

N_i can decrypt this message and obtain the seeds of its neighbors, i.e., S''_{N_k}. Each sensor node can use this method to obtain the seeds of its neighboring sensor nodes for securely communicating with them. Note that if N_i do not send a *join* message to the corresponding cluster head ch_i for becoming a member of the cluster in advance, ch_i will reject its *request* message once N_i wants to obtain the seeds of its neighbors.

The secure communication phase

Once the sensor node discovery phase is finished, each sensor node/cluster head can securely broadcast/multicast data to its neighbors/members using the seeds of its neighbors/members. For example, sensor node N_i wants to broadcast/multicast data M to its neighbors, e.g., N_k. First, N_i randomly chooses a prime RP_{N_i}, a number X_{N_i}, and a session key K_{N_i}. These steps are like the method mentioned before. Second, a multicast group $U_m = \{N_1, N_2, \ldots, N_k\}$ is decided by N_i. This broadcasting message can be explained as follows:

$$ID_{N_i}, \text{location}_{N_i}, B, X_{N_i}, E_{K_{N_i}}(M),$$

where $B = (RP_{N_i} \prod_{N_k \in U_m} H(S''_{N_k} \| X_{N_i})) + K_{N_i}$. After receiving this message, N_k first computes $S''_{N_k} = H(S'_{N_k} \| ID_{N_i} \| \text{Location}_{N_i})$, where $S'_{N_k} = H(S_{N_k} \| ID_{\text{ch}_i} \| \text{Location}_{\text{ch}_i})$. Then N_k can obtain K_{N_i} via computing the following equation:

$$K_{N_i} = B \bmod H(S''_{N_k} \| X_{N_i}).$$

After obtaining K_{N_i}, N_k can decrypt this message and obtain M. N_i can communicate with one or more neighboring sensor nodes in the same manner.

Re-clustering after cluster head capture/compromise

Every cluster head may be compromised or captured in the sensor network by adversaries. The sensor nodes that belong to a compromised cluster head have to be redistributed into new cluster heads. We assume that there is an appropriate intrusion detection system (IDS) used at the base station and cluster heads. The base station can monitor all cluster heads, and each cluster head can also monitor all its members in its cluster. If any cluster head (or sensor node) is compromised, then failure can be detected by the base station (or cluster heads). This assumption is similar to the [2,8]. In this section, we describe the re-clustering scheme step by step. For example, adversaries compromises a cluster head, denoted as ch_c, and obtains all secret information from ch_c, e.g., the seeds S'_{N_i} of its members, in this cluster. All the members of this cluster controlled by ch_c have to be redistributed into new cluster heads. Once the compromised ch_c is detected by the base station, the base station has to revoke ch_c. First, the base station records the sensor nodes of this cluster as a $list_c$, say orphaned sensor nodes [8]. Second, the base station randomly chooses a prime RP_{BS}, a number X_{BS}, and a session key K_{BS}. Third, it has to announce the compromised ID and location of ch_c via sending the following message to all legal cluster heads CH_j in the sensor network:

$$ID_{\text{BS}}, B, X_{\text{BS}}, E_{K_{\text{BS}}}(\text{event}\{ID_{\text{ch}_c}, \text{Location}_{\text{ch}_c}, \text{re-clustering}\}),$$

where $B = (RP_{\text{BS}} \prod_{\text{CH}_j \in U_m} H(S'_{\text{CH}_j} \| X_{\text{BS}})) + K_{\text{BS}}$, and $S'_{\text{CH}_j} = H(S_{\text{CH}_j} \| ID_{\text{BS}})$. After receiving this message, CH_j can decrypt it and knows which cluster head is compromised by adversaries. If CH_j (one or more) is located around ch_c, it will rebroadcast a re-clustering message that contains ($< ID_{\text{CH}_j}, \text{Location}_{\text{CH}_j} >, < ID_{\text{ch}_c}, \text{Location}_{\text{ch}_c} >$) to nearby sensor nodes using the maximum power with a random delay for redistributing the sensor nodes that belong to ch_c into new cluster heads. If a sensor node which receives many re-clustering messages belongs to ch_c, it will need to choose a new cluster head whose re-clustering message has the strongest signal to join the cluster. The following steps are similar to the sensor node join phase and the sensor node discovery phase as mentioned before. Note that the $list_c$ stored in the base station can prevent the false or illegal sensor nodes from joining new clusters during the sensor node join phase. In our re-clustering scheme, ch_c do not possess the private seeds which its members possess. The base station only needs to regenerate the corresponding seeds, i.e., $S'_{N_i} = H(S_{N_i} \| ID_{\text{CH}_j})$, and sends them to CH_j that is located around ch_c. The compromise of the cluster head ch_c can not cause the entire compromise of its cluster.

Revocation after sensor node capture/compromise

When a sensor node, say N_c, is compromised by an adversary, the corresponding cluster head, say ch, has to revoke N_c for avoiding the compromise of future messages. In other words, cluster head ch has to inform its members of the compromise of N_c via broadcasting the following message:

$$ID_{\text{ch}}, \text{Location}_{\text{ch}}, B, X_{\text{ch}}, E_{K_{\text{ch}}}(ID_{N_c} \| \text{Location}_{N_c}),$$

where $B = (RP_{\text{ch}} \prod_{N_i \in U_m} H(S'_{N_i} \| X_{\text{ch}})) + K_{\text{ch}}$. After receiving this message, the members of the cluster controlled by ch can decrypt it and know the compromised sensor node's ID and location. If a sensor node is one of N_c' neighbors, say N_k, it has to remove the corresponding seed, i.e.,

S''_{N_c}, and update the relation with the compromised sensor node N_c. Note that the compromised sensor node N_c does not possess the private seeds of its neighbors N_k, i.e., S_{N_k}, and it only possesses the given seeds of its neighbors from ch, i.e., $S''_{N_k} = H(S'_{N_k} \| ID_{N_c} \| \text{Location}_{N_c})$. N_k can revoke N_c by memorizing the revoked ID_{N_c} only.

Adding new sensor nodes

Sensor nodes may be compromised or exhaust their batteries, so adding new sensor nodes is a critical issue after some running or operation time. Each new sensor node is preloaded with two parameters: $(ID_{\text{new}}, S_{\text{new}})$, an encryption algorithm $E(*)$, and an one-way hash function $H(*)$. After new sensor nodes are randomly deployed, they have to be distributed into new cluster heads. The base station asks each cluster head to rebroadcast a *hello* message for clustering. The follow-up processes are similar to the clustering phase, the sensor node join phase, and the sensor node discovery phase. Note that old sensor nodes may receive *hello* message(s) from one or more cluster heads, they will ignore the message(s). We also assume, like the scheme [16], that the *hello* messages broadcast of sensor nodes is performed during the sensor node discovery phase wherein all sensor nodes are free from compromise. Each sensor node can finish the discovery phase successfully in the process.

Security analysis

Eavesdropping and injection attack

Our proposed method can prevent external adversaries from eavesdropping normal messages or injecting bogus data into the sensor network. Because adversaries do not have the corresponding seeds of sensor nodes, they can not decrypt messages or impersonate a legitimate sensor node to forge messages for disrupting the sensor network.

Sensor node replication attack

Adversaries can deploy malicious sensor nodes which are clones of a compromised sensor node, say A, into multiple locations in the sensor network. There are two scenarios. The first scenario is that a clone is deployed at one location distant from A's original location in the same cluster as A. This will be detected by the corresponding cluster head if the clone sends a *join* message to the cluster head. The second scenario is that a clone is deployed in the different cluster from A. The base station can be aware of which cluster the clone wants to join during the sensor node join phase because it knows each member's ID and the corresponding location of each cluster if A has joined a cluster at one location before. Once the base station knows that the clone of a compromised sensor node may be deployed in the vicinity of a certain cluster head, it can reject the clone's *join* message. Therefore, the base station can make a judgment that A is a compromised sensor node and then takes the appropriate action in order to revoke A.

Sybil attack

Newsome et al. [26] and Zhou et al. [15] were introduced the Sybil attack. In this attack, a malicious sensor node claims multiple IDs or locations. Suppose that a malicious sensor node, say A, impersonates a legitimate or illegitimate sensor node, say B. The malicious sensor node A looks like a new deployed sensor node B from the view of the sensor nodes in the vicinity of A. Sybil attack may lead to many serious effects in sensor network, e.g., inconsistence of the network routing information [22]. Our scheme can defense against Sybil attack because the malicious sensor node A do not possess the corresponding private seed of B. Thus, the malicious sensor node can not successfully impersonate other nodes to inject forged data or routing information into the sensor network without the corresponding private seeds.

Wormhole attack

In the Wormhole attack, adversaries try to tunnel normal messages between two distinct locations by creating an out-of-band and low-latency channel [15,21,22,27,28]. This attack does not compromise any sensor node, but it may lead to many serious threats, e.g., the chaos of the routing operations [22]. In our scheme, suppose that the channel between two far sensor nodes (they are not neighbors) C and D is created by adversaries in a cluster. A legitimate sensor node, say C, receives a *reply* message from another sensor node, say D, during the sensor node discovery phase, and it can check if the location of D is within its transmission range. If D is not within C's transmission range, C will confirm that D is not one of its neighbors. In another situation, when C receives a message sent from D during The Secure Communication Phase, it can also check if the location of D is within its transmission. If D is not within C's transmission range, C will reject the message sent from D. Assuming that adversaries forge the location of D to be within C's transmission range, C can not decrypt the message sent from D because C will use the location of D to compute the corresponding seed, i.e., $S''_C = H(S'_C \| ID_D \| \text{Location}_D)$ in order to obtain the session key K_D via computing the equation $K_{N_D} = B \bmod H(S''_{N_C} \| X_{N_D})$. Because the location of D is fake, C can not decrypt the message sent from D correctly. Therefore, our scheme can defense against Wormhole attack according to locations. Note that if a malicious sensor node forges its location to communicate with other sensor nodes, in all probability, it will be detected by its neighboring sensor nodes (or cluster head) which have its ID and location. Once the neighbors of the malicious sensor node detect the abnormality of it, they will notify the corresponding cluster head of the event.

Sinkhole attack

The authors in [21,28] pointed out that the Sinkhole attack is a serious attack to wireless sensor network routing protocols. In this attack, compromised or malicious sensor nodes try to attract all the messages from their neighbors by tricking other sensor nodes [21]. In other words, a compromised or malicious sensor node wants to become a relay node for attracting all the messages sent by legitimate sensor nodes. Under such attack, our scheme can withstand Sinkhole attack via checking whether the distance between two locations is within the reasonable transmission range or not. With our scheme, the location information advertisements of neighbors of each sensor node can be authenticated. Assuming that a compromised sensor node forges its own location to trick other sensor nodes, in all probability, this attack will be detected by its neighbors (or cluster head) as the mentioned before.

Perfect forward secrecy

Du et al. [5] used a broadcast key in order to securely broadcast messages among neighboring sensor nodes. In addition, the authors in the previous schemes [2,8] used a communication (or session) key for communications among the sensor nodes in the same cluster. This will causes a problem that once adversaries compromise a sensor node of one cluster, the former messages intercepted and collected from the sensor node by adversaries can be decrypted using the communication key of the compromised one. However, our scheme can withstand such situation via changing the session key every time. We recall that after cluster heads or sensor nodes use the parameters, e.g., RP, X, and K, they will erase these parameters from their own memory immediately. This characteristic will ensures that our scheme can achieve perfect forward secrecy. For example, we assume that an adversary has intercepted and collected all messages from a compromised sensor node, say E. The adversary can not decrypt these messages via using the keying material of E. Furthermore, our scheme has another advantage. Once a sensor node is compromised, other legal sensor nodes have to remove the compromised keys in the previous scheme [2,5,8], but we do not need to do so. In Table 1, we compare our scheme with [2,5].

Table 1 Comparison of perfect forward secrecy for three schemes

	AP[5]	SECK[2]	Our scheme
Perfect forward secrecy	No	No	Yes

Performance evaluations

Storage overhead

Each cluster head has to store the seeds, IDs and locations of all its members in order to securely broadcast messages to its members. Because cluster heads are resource-rich, this storage overhead is acceptable for them. Each sensor node also has to store the seeds, IDs and locations of all its neighbors. Because the communication range of each sensor node is limited, the number of these neighbors is also restricted.

For simplicity, we only discuss the required keys for a sensor node in this section. In our proposed method, each sensor node N_i has to store its own private seed, e.g. S_{N_i}, and the seeds of its neighboring sensor nodes, e.g. S''_{N_k}. The number of a sensor node's neighbors depends on the network density. Suppose that a sensor node wants to communicate with its n neighbors, and then the sensor node has to store n seeds of its neighbors in order to securely broadcast/multicast messages. We make a comparison with the previous schemes [2,5] used in a hierarchical sensor network in terms of the required keys.

Du et al. [5] proposed an asymmetric predistribution key management scheme (AP). We recall that a large key pool and the corresponding key IDs are generated at the beginning, and each L-sensor is loaded with l keys, and each H-sensor (e.g., cluster head) is loaded with $M(M \gg l)$ keys without replacement from the key pool. Each sensor node also can set up broadcast keys in order to securely broadcast messages to its neighbors. Assuming that the number of neighbors of a sensor node is n, the sensor node will have to store $n + 1$ broadcast keys in its memory.

Chorzempa et al. [2], the authors proposed the SECK. Prior to deployment, each sensor node has to store the complete administrative keys $\{K_{a_1}, K_{a_2}, \ldots, K_{a_{k+m}}\}$ and a pairwise secret key K_{p_i} shared with the base station. After deployment, each sensor node is required to store only a subset of the administrative keys $\{K_{a_1}, K_{a_2}, \ldots, K_{a_{k+m}}\}$, i.e., k keys, and one tree administrative key K_{t_i} assigned by its AFN (cluster head). Table 2 offers a view of storage overhead for three schemes.

Communication overhead

Mica2 motes are widely used in wireless sensor networks, and we use the following consumption rates [29]: 16.25 and 12.25 μJ/byte for transmission and reception to Mica2 motes, respectively. We also assume that ID, location and X are 2, 2, and 8 bytes, respectively. The communication overhead of the sensor node discovery phase is evaluated using the assumptions mentioned above. This phase

Table 2 Comparison of storage overhead for three schemes in terms of the number of keys stored in each sensor node

	AP[5]	SECK[2]	Our scheme
Prior to deployment	l	$k + m + 1$	1
After deployment	$l + n + 1$	$k + 2$	$n + 1$

incurs the following communication cost. We recall that when a sensor node N_i tries to find its one-hop neighbors within its transmission range, it broadcasts a *hello* message that contains its $(ID_{N_i}, \text{Location}_{N_i}, ID_{\text{ch}_i})$, which is 6 bytes, to its one-hop neighbors. The energy consumption of the payload for transmission and reception is 97.5 and 73.5 μJ, respectively. Assuming that a sensor node has n neighbors, its communication overhead is (97.5+73.5n) μJ. This time complexity of communication overhead is $O(n)$ according to the number of neighbors of one sensor node.

In the following, we evaluate the communication overhead of one sensor node according to the number of a multicast group during the secure communication phase. This communication overhead of broadcasting/multicasting messages depends on the size of a multicast group U_m. In our scheme, the additional communication cost of transmission and reception is acceptable for resource-constrained sensor nodes during the secure communication phase because the number of neighbors of every sensor node is limited by small transmission range. For example, assuming that sensor node N_i wants to broadcast/multicast data M to its neighbors as the mentioned in Section "The proposed method", it has to send the following message: ID_{N_i}, location$_{N_i}$, B, X_{N_i}, and $E_{K_{N_i}}(M)$. The bigger the quantity of U_m of sensor node N_i is, the bigger the value of B is, e.g., $B = (RP_{N_i} \prod_{N_k \in U_m} H(S''_{N_k} \| X_{N_i})) + K_{N_i}$. The time complexity of B is $O(n^2)$ according to the number of neighbors of one sensor node.

Computation overhead

In our scheme, we do not employ any public key technique for communications among nodes, instead we use symmetric cryptography, multiplication and mod operations to encrypt/decrypt data and compute B and K, respectively. These operations are not a big computation overhead used in resource-constrained sensor nodes. Assuming that sender N_i transmits a message $(ID_{N_i}, \text{location}_{N_i}, B, X_{N_i}, E_{K_{N_i}}(M))$, in which B is computed by N_i using multiplication operation, to receiver N_k, and then the receiver N_k can obtain K_{N_i} from the message sent by sender N_i using mod operation to compute the following equation: $K_{N_i} = B \bmod H(S''_{N_k} \| X_{N_i})$. After obtaining K_{N_i}, the receiver can use this key K_{N_i} to decrypt the message via the symmetric cryptography which is suitable for resource-constrained sensor nodes. We evaluate energy cost of symmetric-key and hash algorithms using [30]. We use the following assumption rates: 1.62/2.49 μJ/byte and 5.9 μJ/byte for AES with 128-bit keys for data encryption/decryption and SHA-1 for hashing, respectively. The energy cost of encrypting/decrypting a 20 bytes data and hashing a 136 bytes data are 32.4/49.8 μJ and 802.4 μJ, respectively. As reported in [14], the modular inverse computation and modular exponentiation operation are the most time-consuming operations. Therefore, our scheme do not use these two operations, and we only use the multiplication and mod operations for sensor nodes.

Conclusion

We propose a secure broadcast/multicast scheme for hierarchical sensor networks. Each node is only preloaded with one private seed prior to deploy, and the memory size can be minimal for resource-constrained sensor nodes. In our method, the revocation of compromised sensor nodes or cluster heads becomes easier than the previous schemes which need the operation of rekeying because each sensor node or cluster head does not possess the private seeds of its neighbors. The resource-rich cluster heads are responsible for the management and distribution of seeds for their members. Our scheme can defense against the common attacks of wireless networks. Changing the session key every time can also achieve perfect forward secrecy in our scheme. Adversaries can only intercept and collect the former messages of one cluster, but they can not decrypt these messages once a certain sensor node of the cluster is compromised.

Competing Interest

The authors declare that they have no competing interests.

Acknowledgements

This work was partially supported by the National Science Council, Taiwan, under contract no. NSC100-2410-H-005-046. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

Author details

[1]Department of Distribution Management, National Chin-Yi University of Technology, Taichung, Taiwan. [2]Department of Management Information Systems, National Chung Hsing University, Taichung, Taiwan. [3]Department of Photonics and Communication Engineering, Aisa University, Taichung, Taiwan.

Received: 4 December 2011 Accepted: 6 July 2012
Published: 6 August 2012

References

1. H Chan, A Perrig, D Song, Random key predistribution schemes for sensor networks. in *Proceedings of the 2003 IEEE Symposium on, Security and Privacy* (The Claremont Resort Oakland, California, USA, 11-14 May 2003), pp. 197–213
2. M Chorzempa, JM Park, M Eltoweissy, Key management for long-lived sensor networks in hostile environments. Comput. Commun. **30**(9), 1964–1979 (2007)
3. W Du, J Deng, YS Han, PK Varshney, A pairwise key predistribution scheme for wireless sensor networks. in *Proceedings of the 10th ACM Conference on Computer and Communications (SecurityCCS'03)* (Washington, DC, USA, 27-30 October 2003), pp. 42–51
4. W Du, J Deng, YS Han, PK Varshney, A Khalili, A pairwise key predistribution scheme for wireless sensor networks. ACM Trans. Inf. Syst. Secur. **8**(2), 228–258 (2005)

5. X Du, Y Xiao, M Guizani, HH Chen, An effective key management scheme for heterogeneous sensor networks. Ad Hoc Netw. **5**(1), 24–34 (2007)
6. M Eltoweissy, H Heydari, L Morales, H Sudborough, Combinatorial optimizations of group key management. J. Netw Syst. Manage. **12**(1), 30–50 (2004)
7. L Eschenauer, VD Gligor, A key management scheme for distributed sensor networks. in *Proceedings of the 9th ACM Conference on Computer and Communication Security (CCS'02)* (New York, NY, USA, 2002), pp. 41–47
8. MF Younis, K Ghumman, M Eltoweissy, Location-aware combinatorial key management scheme for clustered sensor networks. IEEE Trans. Parallel Distrib. Syst. **17**(8), 865–882 (2006)
9. MY Hsieh, YM Huang, HC Chao, Adaptive security design with malicious node detection in cluster-based sensor networks. Comput. Commun. **30**(11-12), 2385–2400 (2007)
10. D Liu, P Ning, Multilevel μTESLA: broadcast authentication for distributed sensor networks. ACM Trans. Embed. Comput. Syst. **3**(4), 800–836 (2004)
11. A Perrig, R Szewczyk, JD Tygar, V Wen, DE Culler, SPINS: security protocols for sensor networks. Wirel. Netw. **8**(5), 521–534 (2002)
12. WR Heinzelman, A Chandrakasan, H Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks. in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS)* (Island of Maui, 4-7 January 2000), pp. 3005–3014
13. LB Oliveiraa, A Ferreirac, MA Vilaca, HC Wong, M Bern, R Dahab, AAF Loureiro, SecLEACH-On the security of clustered sensor networks. Signal Process. **87**(12), 2882–2895 (2007)
14. HF Huang, A novel access control protocol for secure sensor networks. Comput. Stand. Interfaces. **31**(2), 272–276 (2009)
15. Y Zhou, Y Zhang, Y Fang, Access control in wireless sensor networks. Ad Hoc Netw. **5**(1), 3–13 (2007)
16. S Zhu, S Setia, S Jajodia, LEAP+: efficient security mechanisms for large-scale distributed sensor networks. ACM Trans. Sens. Netw. **2**(4), 500–528 (2006)
17. A Shamir, Identity-based cryptosystems and signature schemes. in *Proceeding of the Cryptology-Crypto'84* (Santa Barbara, California, USA, 19-22 August 1984), pp. 47–53
18. M Bellarea, C Namprempre, G Neven, Security proofs for identity-based identification and signature schemes. in *Proceeding of the EUROCRYPT'04* (Interlaken, Switzerland, 2-6 May 2004), pp. 268–286
19. CP Schnorr, Efficient signature generation for smart card. J. Cryptol. **4**(3), 161–174 (1991)
20. Xi Cao, L Dang, W Kou, B Zhao, IMBAS: identity-based multi-user broadcast authentication in wireless sensor networks. Comput. Commun. **31**(4), 659–667 (2008)
21. Y Zhang, W Liu, Y Fang, D Wu, Secure localization and authentication in ultra-wideband sensor networks. IEEE J. Sel. Areas Commun. **24**(4), 829–835 (2006)
22. Y Zhang, W Liu, W Lou, Y Fang, Location-based compromise-tolerant security mechanisms for wireless sensor networks. IEEE J. Sel. Areas Commun. **24**(2), 247–260 (2006)
23. S Capkun, J-P Hubaux, Secure positioning of wireless devices with application to sensor networks. in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies,* vol. 3 (Miami, FL, USA, 13-17 March 2005), pp. 1917–1928
24. A Savvides, C Han, M Strivastava, Dynamic fine-grained localization in ad hoc networks of sensors. in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking* (Rome, Italy, 2001), pp. 166–179
25. CC Chang, YW Su, IC Lin, A broadcast-encryption-based key management scheme for dynamic multicast communications. in *Proceedings of the 2nd International Conference on Scalable Information Systems* (Suzhou, China, 6-8 June 2007)
26. J Newsome, E Shi, D Song, A Perrig, The sybil attack in sensor networks: Analysis & defenses. in *Proceedings of The 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)* (Berkeley, California, USA, 2004), pp. 26–27
27. Y Hu, A Perrig, D Johnson, Packet leashes: a defense against wormhole attacks in wireless ad hoc networks. in *Proceedings of the 22th Annual Joint Conference of the IEEE Computer and Communications Societies,* vol. 3 (San Francisco, CA, 2003), pp. 1976–1986
28. C Karlof, D Wagner, Secure routing in wireless sensor networks: attacks and countermeasures. Ad Hoc Netw. **1**(2), 293–315 (2003)
29. LB Oliveira, HC Wang, AA Loureiro, LHA-SP: Secure protocols for hierarchical wireless sensor networks. in *Proceedings of 9th IFIP/IEEE International Symposium on Integrated Network Management* (Nice, France, 15-19 May 2005), pp. 31–44
30. A Wander, N Gura, H Eberle, V Gupta, S Shantz, Energy analysis of public-key cryptography for wireless sensor networks. in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications* (Kauai Island, HI, USA, 8-12 March 2005), pp. 324–328

doi:10.1186/1687-1499-2012-241
Cite this article as: Chen *et al.*: Providing perfect forward secrecy for location-aware wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:241.