Volume 30, issues 11–12, 10 September 2007 ISSN 0140-3664

computer
communications

Available online at
ScienceDirect
www.sciencedirect.com

Special Issue:
Security on Wireless Ad Hoc and Sensor Networks
Guest Editors: Sghaier Guizani
Hsiao-Hwa Chen
Peter Mueller

Available online at www.sciencedirect.com

ScienceDirect

Computer Communications 30 (2007) 2365–2374

computer communications

www.elsevier.com/locate/comcom

Pair-wise path key establishment in wireless sensor networks

Jang-Ping Sheu *, Jui-Che Cheng

Department of Computer Science and Information Engineering, National Central University, Jhongli 32054, Taiwan, ROC

Available online 5 May 2007

Abstract

When sensor networks deployed in unattended and hostile environments, for securing communication between sensors, secret keys must be established between them. Many key establishment schemes have been proposed for large scale sensor networks. In these schemes, each sensor shares a secret key with its neighbors via preinstalled keys. But it may occur that two end nodes which do not share a key with each other could use a secure path to share a secret key between them. However during the transmission of the secret key, the secret key will be revealed to each node along the secure path. Several researchers proposed a multi-path key establishment to prevent a few compromised sensors from knowing the secret key, but it is vulnerable to stop forwarding or Byzantine attacks. To counter these attacks, we propose a hop by hop authentication scheme for path key establishment to prevent Byzantine attacks. Compared to conventional protocols, our proposed scheme can mitigate the impact of malicious nodes from doing a Byzantine attack and sensor nodes can identify the malicious nodes. In addition, our scheme can save energy since it can detect and filter false data not beyond two hops.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Byzantine attacks; Path key establishment; Security; Wireless sensor networks

1. Introduction

Security is an important requirement in many sensor network applications, especially in unattended and often hostile environments such as battlefield surveillance, friendly forces monitoring, and biological attack detection. Since wireless sensor networks are much easier for an adversary to eavesdrop any packet transmitted on the channel, it is necessary for two neighboring nodes to share a secret key to encrypt sensitive data and authenticate peer-to-peer communication. Furthermore, sensor nodes are typically small battery-equipped devices with very limited communication, computation, and memory capacity. Traditional key establishment techniques like public key cryptosystem (e.g. RSA) [20,23] are impractical.

One practical solution is key pre-distribution, in which keys have to be installed onto sensors before deployment so that nodes can use shared keys to conduct secure communication. Using the key pre-distribution scheme to establish a secret key has two extreme examples. One example is where each sensor node is preloaded with $N-1$ pair-wise keys before deployment, such that it shares with any node a secret key, where N is the number of nodes in the networks. This scheme offers the most security since no key information will be known between sensor pairs from a compromised node. However, this scheme is not suitable for large networks, since a sensor may need to store thousands of keys, which increase linearly with network size. The other extreme example is where all sensor nodes use the same master key in the network. The advantage of this scheme is that a sensor node needs only a master key regardless of the network size. However, this scheme suffers low security, since if one of the sensor nodes is compromised, the communication of the entire network will be known.

To overcome the disadvantage of the above schemes, several key pre-distribution schemes have been recently proposed [1–10]. The Random Key Pre-distribution scheme (RKP) was first proposed in [1] for large-scale sensor networks. In this scheme, each node randomly picks m keys from a large key pool, such that any two sensor nodes will share at least one common key with a certain

* Corresponding author. Tel.: +886 3 4227151.
E-mail address: sheujp@csie.ncu.edu.tw (J.-P. Sheu).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.04.021

probability. The q-composite key pre-distribution scheme [2] requires two sensors that share at least q ($q > 1$) pre-distributed keys in order to establish a common key. This improves resilience against node capture attacks. However, both schemes [1,2] are vulnerable to nodes compromise attack because a small number of compromised nodes may expose a large fraction of common keys between the non-compromised nodes. The Threshold-Based Key Predistribution (TBKP) techniques were developed in [3,4] to improve [1,2] drawback. After the sensor nodes are deployed, a unique pair-wise key can be established between any pair of neighboring nodes. When the number of compromised sensors becomes less than a threshold, any other keys shared between non-compromised sensors will not be affected. Key pre-distribution schemes based on knowing sensor deployment knowledge were proposed in [5–9]. This deployment knowledge can further reduce memory requirement of sensor nodes and enhance network resilience against node compromises. The PIKE scheme proposed in [10] addressed the problem of high density deployment requirements in RKP and TBKP. Multi-path pair-wise key establishment protocols [2,11–13,19] were proposed to enhance path-key establishment security by preventing compromised sensor nodes on the single communication path from knowing the established pair-wise key.

The path key exposure problem was introduced in [8]. The problem described a scenario when nodes without a common key to other nodes in the network are required to establish a key through a secure path. So a path key will be used to establish a secret key between two nodes, path key means that key is transmitted using secure communication channel through one or more sensor nodes. However, a secret key may be exposed if one of the nodes along the path is compromised. Some multi-path key establishment protocols were proposed in [2,11–13,19] to solve the path key exposure problem. Establishing a path key by multiple secure paths can significantly decrease the risk of the path key being revealed between a source node and a destination node. But these schemes still have some drawbacks. Since adversaries can launch various inside attacks, they can compromise sensor nodes in any one of multi-paths. For examples, they might alter, spoof, or drop information to disrupt the normal operation of the sensor network. Moreover, adversaries may inject bogus data into the network to consume scarce network resources. So far, these proposed schemes cannot detect and identify the malicious behavior nodes.

In this paper, we propose a pair-wise path key establishment scheme through a multi-path approach to improve the security of path key establishment. In our scheme, a secret key K is partitioned into m key segments by a source node and then is sent to a destination through n node-disjoint paths. A destination node can receive enough key segments to reconstruct the secret key K. So even if a small number of nodes are compromised, the secret key as a whole is not compromised. Besides, a hop by hop authentication method is used to detect false key segments and malicious nodes in each node disjoint path. We designed this scheme to filter bogus traffic injected by adversaries during early transmission stages and to save precious energy. We show through analysis that our scheme is highly secure against node capture, and outperforms other schemes when preventing various attacks in wireless sensor networks.

The rest of this paper is organized as follows. Section 2 introduces related work and background knowledge used in the paper. Section 3 presents our protocols. Section 4 compares the performance of our protocol with previous work. Section 5 concludes this paper.

2. Related work

Recently, path key establishment schemes, which send key segments with multiple paths have been proposed in [12,13,19]. These schemes use multiple physical paths or logical paths forwarding. This way, they aim to reduce the secret key from being known when a compromised node is on the paths. However, when a malicious node modifies or stops forwarding the key value, these schemes fail to obtain the original value. The authors in [12] proposed end-to-end pair-wise key establishment using multiple node-disjoint paths. Assuming that node u needs to set up a pair-wise key with another node v. Firstly, node u finds n node-disjoint paths to v. Secondly, node u selects a key K and divides it into n segments, such that $K = K_1 \cup K_2 \cup K_3 \cup \cdots \cup K_n$. Then node u sends K_i through the ith secure path to node v. When node v receives all n segments of the key K, it can reproduce the key K and use it to secure communication with node u.

Another end-to-end key establishment scheme was proposed in [13]. Assuming that node u wants to set up a path key to node v. Firstly, u sends out its key ID list to v. Secondly, v constructs a key K and cuts it apart into n segments $K_1, K_2, \ldots,$ and K_n. Then v broadcasts requested-to-proxy packets containing the key ID lists of node u and node v to the network. Each proxy node examines the key ID lists to see if it shares keys with both u and v. If the proxy node shares secret keys with nodes u and v, it responds to v with a share key ID used to communicate with v. If the proxy node does not share any keys with nodes u and v or if it has received the same packet request from v, it forwards this request to a random neighbor other than the sender. When node v receives n replies from proxies, it transmits the n key segments to u via the n proxies with secure communications. When node u receives all n segments, node u can reconstruct the original key K. This scheme uses multiple proxies to help against node capture as the secret key is transmitted between the source node and destination node. However, it may transmit secret shares on the same physical path and there still exists the risk if the captured node sits on the intersect point of several paths between these proxies and drops all the key shares passing through it. Moreover, it cannot prevent stop forwarding attacks and it cannot detect malicious behavior nodes.

In [19] the authors proposed a scheme that does not require a source node to discover n node-disjoint paths to the target node. It is used to add edges in the key graph such that there are n logical paths between source and target nodes. After that, key establishment is straightforward. The drawback of this scheme is that if the two nodes which want to establish a key are too far from each other, the intermediate nodes need to recursively establish more temporary keys. This method adds a lot of extra communication overhead and energy consumption. Furthermore, when the source node sends key segments, routing will use the same physical path to send the key shares. So a malicious node on the path can drop the packets, and this protocol cannot prevent the stop forwarding and Byzantine attacks [22].

The authors in [10] used a key pre-distribution scheme called PIKE to achieve the path key establishment. PIKE can guarantee that any two nodes in a network always share a key with an intermediary node. This intermediary node is then used to establish a path key between the two nodes. But this approach makes a large fraction of neighboring sensor pairs that do not share preloaded keys, and thus they need to establish path keys. Consequently, the PIKE scheme involves a relatively high communication overhead, making it unsuitable for large sensor networks. In [14] the authors proposed a group-based key establishment scheme to reduce the communication overhead of the scheme proposed in [10]. However, in both schemes the secret key does not split up, so the intermediary nodes may be able to know the secret key.

All the above schemes are vulnerable for stop forwarding and Byzantine attacks. When a compromised node alters the key shares on the path, the false secret share will not filter and will continue forwarding to the destination. This may cause energy consumption on the path, and the receiver might get false shares. Moreover, they can not detect malicious nodes, so a malicious behavior node can do anything on the network.

In this paper, we adopt the secret sharing scheme [15] and one-way hash chains [24] to help our scheme against the above attacks. A (t, n) threshold secret sharing scheme is a method of dividing a secret s into n shares $s_1, s_2, \ldots, s_n$, such that the knowledge of any t or more shares makes the secret s easily computable, but knowledge of $t-1$ or less shares does not reveal any information about the secret s, where t is the threshold of the scheme. It is possible that, after compromising a node, the adversary may attempt to cheat our system by sending us faked or altered message shares. The secret sharing scheme has the ability of cheater detection and identification in secret share. By embedding the cheater detection and identification scheme [17], we can avoid reconstructing false message. One-way hash chains are used to generate a key chain of length $n+1$, where the first element of the chain h_0 is randomly picked, and the chain is generated by repeatedly applying a one-way hash function H. The function H should be simple to compute but must be computationally infeasible to invert in general. Any key h_j can be verified from h_i $(1 \leqq i < j \leqq m)$ to be indeed an element in the chain by repeatedly applying H for $j-i$ times, so that $h_j = H^{j-i}(h_i)$.

3. Pair-wise path key establishment protocol

This section presents our path key establishment scheme. Our protocol is aimed to prevent active attacks such as the stop forwarding and Byzantine attacks that adversaries may use the compromised nodes to alter messages and prevent the key establishment. We need an authentication mechanism to prevent the active attacks. Our protocol consists of two phases. The first phase runs a group-based key pre-distribution scheme. The nodes are partitioned into a number of groups. The nodes within a same group have shared unique pair-wise keys and a fraction of pair-wise keys with their neighboring groups. Thus, pair-wise key establishment between two neighboring sensor nodes require only checking if they are from the same group, and locally establish a key when they are in neighboring groups. After the end of this phase, an arbitrarily pair of neighboring nodes in the network will have a high probability for sharing a secret pair-wise key. The second phase does the end to end pair-wise key establishment if a pair of nodes is a multi-hop away and wants to establish a share key. Firstly, the two communicating end nodes establish a number of node-disjoint paths. Secondly, each node establishes a separate secret shared key with its 2-hop neighbors in the path. Thirdly, the source node disperses a pair-wise key into a number of pieces and sends them to different paths to the destination node. After, we use the hop by hop authentication method to prevent Byzantine attacks and use the scheme like TWOACK [17] to prevent stop forwarding attacks.

The presented protocol can detect a malicious node when it alters the data and discard the false data, which can avoid exhaust the precious energy of relaying nodes on any forwarding path. Our key establishment method has perfect resilience against node compromises since the pair-wise keys are built between two individual nodes. No matter how many nodes are compromised, adversaries will know nothing about the keys shared between non-compromised nodes. Thus, the non-compromised node links always remain secure. If we use a unique pair-wise key between two neighboring nodes, we can also use the pair-wise key as a node to node authentication [2] method. This ensures that the compromised node cannot impersonate its neighbors to other nodes.

3.1. Key pre-distribution scheme with deployment knowledge

In the following, we describe our key pre-distribution scheme with deployment knowledge. The deployment knowledge can tell us what local area a sensor node is more likely to appear in and what sensor nodes are more likely to be its neighbors. With the help of deployment knowledge, we can achieve a higher degree of connectivity with a lower memory requirement. In this paper, we assume that N nodes are uniformly deployed to an arbitrary two-dimensional sensing field S_f. This sensing field can be divided into $g = a \times b$ small hexagonal grids. Let each small grid denote as $z(x,y)$, for $1 \leqq x \leqq a$ and $1 \leqq y \leqq b$. Let $G_{(x\ y)}$ denote as a group of sensors randomly deployed in grid $z(x,y)$. Each group has $c = N/g$ sensors. The group $G_{(x,y)}$ includes sensors with IDs from $c((x-1)b+y-1)+1$ to $c((x-1)b+y)$. $N = \bigcup_{(x,y)=(1,1)}^{(a,b)} G_{(x,y)}$. For example, if each group has 100 sensor nodes, $a = 10$, and $b = 5$, the group $G_{(2,3)}$ has the sensor nodes with IDs from 701 to 800. Fig. 1 shows a number of sensor nodes uniformly distributed in 12 hexagonal grids with deployment knowledge. In our scheme, we partition a sensor field into hexagonal grids and design our key pre-distribution with hexagonal deployment knowledge. To maintain the same key connectivity, the key pre-distribution with deployment knowledge should have less pair-wise keys in each node than that of the scheme without deployment knowledge. Using the hexagonal deployment knowledge to design the key pre-distribution scheme idea first is used in [7,8].

Before the sensor nodes are deployed, we preload each pair of sensors in the same group with a unique pair-wise key. Thus, each sensor is required to be preloaded with $c - 1$ pair-wise keys shared with the same group. Besides, each sensor node in a group has probability p_s which has a pair-wise key with the sensor nodes in its adjacent groups. Therefore, each sensor node needs to preload $6 \cdot c \cdot p_s$ pair-wise keys shared with the sensors in its adjacent groups. Furthermore, we use the method in [10] to reduce the memory requirement by a factor of two, i.e., the total memory overhead per sensor is $M = \lceil \frac{1}{2}(c-1) \rceil + \lceil \frac{1}{2}(6 \cdot c \cdot p_s) \rceil$. For example, assume there are 10^4 nodes uniformly deployed to 100 hexagonal grids over a $10^3 \times 10^3\ \text{m}^2$ area and $p_s = 0.5$. Then each node needs to store $\lceil \frac{1}{2}(100-1) \rceil + \lceil \frac{1}{2} 600 \cdot 0.5 \rceil = 200$ pair-wise keys. Based on the above key pre-distribution scheme, a group of nodes are deployed in a small local area such that most neighbors of a node come from its own group or neighboring groups and has a high probability of shared keys.

After preloading keys in each node and distributing the nodes in the sensing field, each node broadcasts its ID and Group ID to its one-hop neighbors by searching for shared keys. If two neighboring nodes find that they have no pair-wise keys, they can establish a pair-wise key as follows. Suppose node A has no pair-wise key with its neighbor B, node A can try to establish a pair-wise key with node B as follows. Firstly, nodes A and B exchange the IDs of neighboring nodes which have pair-wise keys with them. Once they identify the common neighbors which share a key with both nodes A and B, a pair-wise key is established through these common neighbors. We name these common neighbors as assistance neighbors. If the number of assistance neighbors $n \geqq 3$, node A generates a path key $K_{A,B}$ with a (t,n) secret sharing mechanism which partitions $K_{A,B}$ into n key segments such that $K_{A,B}$ can be computed from any t of n key segments. A then routes the n key segments through n assistance neighbors to B. When B receives more than t key segments, the path key $K_{A,B}$ can then be constructed by B. After each node has tired to establish a pair-wise key with its neighbors if possible, we can then establish an end-to-end path key in the next subsection.

3.2. Multiple paths end-to-end pair-wise path key establishment

To solve the end-to-end path key establishment problem, we have the following assumptions. Assuming source node S and destination node D are not captured and the

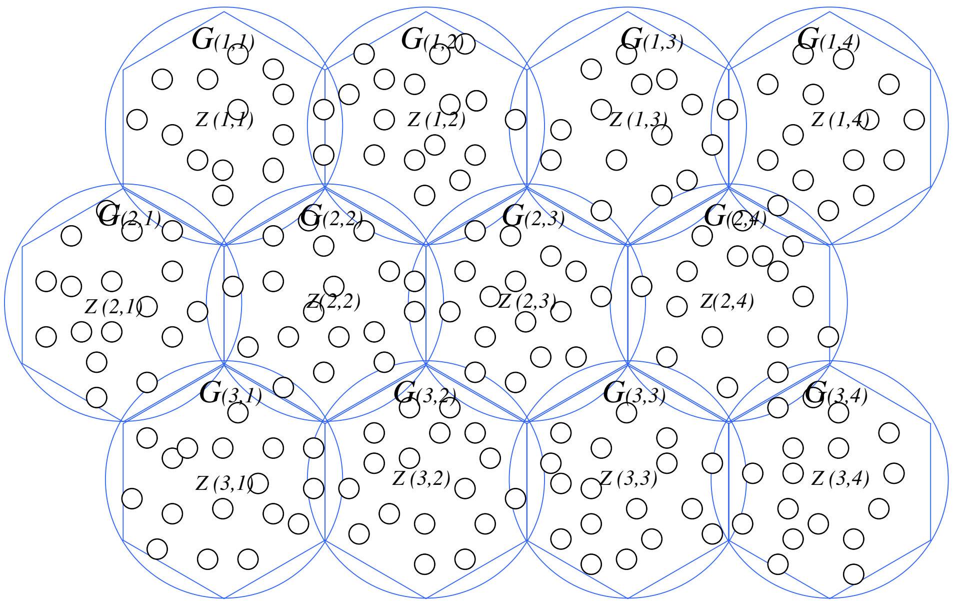

Fig. 1. Group-based key pre-distribution.

communication channels are bidirectional, i.e. if a node u can receive a message from v, it can also send a message to v. We further assume that the sensor nodes are deployed in high density. Here, we consider the following cases for the end-to-end path key establishment. If S and D are in the same group, they can use their pair-wise key as the path key. If S and D belong to different groups which do not share keys with each other, they can establish a path key as follows. S first finds n node-disjoint paths to D. Then S splits the secret pair-wise key K into n segments and sends each segment to different node-disjoint paths with our hop by hop authentication protocol. We describe the details of the end-to-end path key establishment as follows.

Step 1: If node S wants to establish a key with node D, node S must generate a one-way hash key chain before finding the node-disjoint paths. To create a one-way hash chain, S chooses a random value x and computes the list of values $h_0, h_1, h_2, \ldots, h_m$, where $h_0 = x$ and $h_i = H(h_{i-1})$, for $0 < i \leqq m$. We denote $h_1, h_2, \ldots$, and h_m as hash keys. A node-disjoint routing protocol [18] is used to find node-disjoint paths. In this protocol, we include h_m in route request messages (RREQ). Each node may receive many RREQ packets with different values of the nth hash key h_m if there exist a few malicious nodes which can forge false data. The majority rule is used to determine the nth hash key h_m. When the node-disjoint paths between S and D are found [18], the destination node D selects n node-disjoint paths and sends route reply messages (RREP) to different node-disjoint paths. Because the node IDs of the entire path are included in the RREP, each intermediate node of each path receiving an RREP packet will record the next one-hop and next two-hop neighboring nodes in its routing table. Besides, each intermediate node will check to see if it has a pair-wise key with its next two-hop node. If it doest not have a pair-wise key with its next two-hop node, the intermediate node will establish a two-hop path key by using a multiple paths and secret sharing scheme. Note that, this situation will only happen to the neighboring groups.

For example, consider six nodes S, A, B, C, E, and D, which is one of the node-disjoint paths between nodes S and D as shown in Fig. 2. Node D generates a RREP packet that contains the nodes list of the routing path and sends it along the reverse routing path. Each intermediate node, which receives a RREP packet, will record its next hop node and next two-hop node in its routing table. Then each intermediate node can check if it has a pair-wise key with its next hop and next two-hop neighboring nodes.

Step 2: After finding the n node-disjoint paths, the source node divides the path key $K_{S,D} = SK_1 \cup SK_2 \cup \cdots \cup SK_n$. Node S sends each key segment SK_i, $1 \leqq SK_i \leqq n$ through the ith secure node-disjoint path.

Step 3: In each path transmission of key segment to destination D, we use the hop by hop authentication method to ensure that the key segments are correctly received by the destination node.

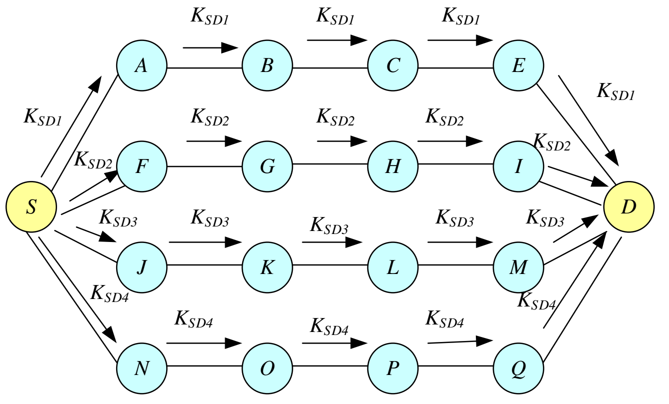

Fig. 2. An example of multi-path key establishment with the (3,4) secret sharing scheme.

Step 4: Upon receiving t or more key segments, node D can rebuild the path key $K_{S,D}$. Thus node D can securely communicate with node S with key $K_{S,D}$. For example, in Fig. 2, there are four secure node-disjoint paths, and node S sends the key shares through these paths. If the (3,4) secret sharing scheme is adopted, node D can rebuild key $K_{S,D}$ when it receives three or four key segments.

The details for the hop by hop authentication scheme are described as follows. We assume a misbehaving node always returns an acknowledgement to its preceding two-hop neighbor when it receives data packets successfully from its two-hop neighbor. If a misbehaving node refuses to send an acknowledgement packet to its preceding two-hop neighbor, our scheme can detect a malicious link. When an intermediate node on a path receives a message, it will not know whether the received message is altered or not. Moreover, we should prevent forged message from forwarding to destination. We can detect and identify those who alter the message by running a malicious node detection and identification phase. The basic idea of malicious node detection is let an intermediate node receive messages from its previous one-hop and two-hop neighbors. After, the intermediate node can verify the consistence of the message sent from its one-hop and two-hop neighbors. If both messages have the same value, the intermediate node will forward the received message to its next one-hop and two-hop nodes. If the received data are different, we can know that one of its preceding one-hop and two-hop nodes is a malicious behavior node, and the data will not be forwarded.

When an intermediate node detects an inconsistent in data received from its one-hop and two-hop neighbors, the intermediate node sends out a key disclosed request to the source node for the authentication of the key segment using a hash key h_{m-1}. The source node does a hash key disclosure procedure and replies the hash key to the intermediate node. When the intermediate node receives the hash key, it will be able to verify who altered the message. Assuming the source node S sends the ith key segment SK_i along the ith node-disjoint path to destination D. Let $K_{i,j} = K_{j,i}$ be the share key of nodes i and j. A *plaintext* encrypted by a key K is denoted as $K\{plaintext\}$. In the malicious behavior node detection phase, we have the following three steps:

Step 1: Assuming nodes A and B are the next one-hop and two-hop neighbors of source node S. Node S sends a key segment SKi through the ith node-disjoint path. To detect a malicious node, the key segment SKi includes the message authentication code (MAC) [25] denoted as $MAC_{h_{m-1}}\{SK_i\}$, which is calculated by the $(m-1)$th one-way hash key h_{m-1}. Node S encrypts SK_i and $MAC_{h_{m-1}}\{SK_i\}$ with pair-wise keys $K_{S,A}$ and $K_{S,B}$ and denoted as $K_{S,A}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ and $K_{S,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$, respectively. Then, node S sends $K_{S,A}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ and $K_{S,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ to nodes A and B, respectively. Note that, each sender uses the MAC mechanism to let the receiver verify whether the message is sent from the sender. It also detects changes in the message. Since node S cannot reach node B directly, node A will relay the messages $K_{S,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ to node B.

Step 2: When node A receives the encrypted message, it will decrypt the message $K_{S,A}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ with pair-wise key $K_{S,A}$. Assuming node C is the next two-hop neighbor of node A, node A encrypts the messages SK_i and $MAC_{h_{m-1}}\{SK_i\}$ using pair-wise keys $K_{A,B}$ and $K_{A,C}$ denoted as $K_{A,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ and $K_{A,C}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$, respectively. Then, node A forwards the messages $K_{A,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ and $K_{S,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ to node B and $K_{A,C}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ through node B to node C.

Step 3: When node B gets the messages from node A, it decrypts the message $K_{A,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ and $K_{S,B}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ with pair-wise keys $K_{A,B}$ and $K_{S,B}$, respectively. If both decrypted messages have the same value of SK_i, node B will encrypt the messages SK_i and $MAC_{h_{m-1}}\{SK_i\}$ with the pair-wise keys of its next one-hop and two-hop nodes and, forward the encrypted messages to them. The above procedures occur continuously until the encrypted messages arrived at the final destination D. However, if the decrypted values of SK_i are different, node B stops forwarding the key segment SK_i and executes the malicious identification phase. In the above scenario, node B can judge if node A is a malicious node directly since source node S is a trust uncompromised node.

In the following, we assume an intermediate node E receiving two different key values SK_i from its preceding one-hop node C and two-hop node B. Node E then executes the following steps to identify which among nodes C and B is the malicious one.

Step 1: Node E sends a key disclosure request $ReqKey$ to source node S through two different paths, the odd path and the even path. The odd (even) path is a path traveling through an odd (even) number of hops back to the source node. An example is shown in Fig. 3(a). The $ReqKey$ message is encrypted by pair-wise keys of senders and receivers. For example, in Fig. 3(a), the $ReqKey$ is encrypted by $K_{E,C}$ and $K_{E,B}$ and sends to nodes C and B, respectively. When node C receives $K_{E,C}\{ReqKey\}$, it decrypts the $K_{E,C}\{ReqKey\}$ message and encrypts it using $K_{C,A}$ then forwards the encrypted $ReqKey$ to its two-hop neighbors A. Note that, a node forwarding a message to its two-hop neighbor need to pass through its one-hop neighbor. Finally, the source node receives two $ReqKey$ messages from the odd and even paths. If there is only one malicious node in a node-disjoint path, the source node eventually will receive at least one correct $ReqKey$ message.

Step 2: When the source node receives $ReqKey$s from the odd and even paths, it then sends the key h_{m-1} through the odd and even paths back to the requested node E. The way of sending key h_{m-1} back to the requested node is same as sending $ReqKey$ to the source node. An example is shown in Fig. 3(b). When node E receives the h_{m-1} key, it checks to see if this key is from the source node by executing function $H(h_{m-1}) = h_m$, where h_m was received in the establishment of node-disjoint paths. If the key is verified as sent from the source node, it can verify the received key segments with hash key h_{m-1}. Firstly, node E computes the MAC on the key segment sent from node B using the

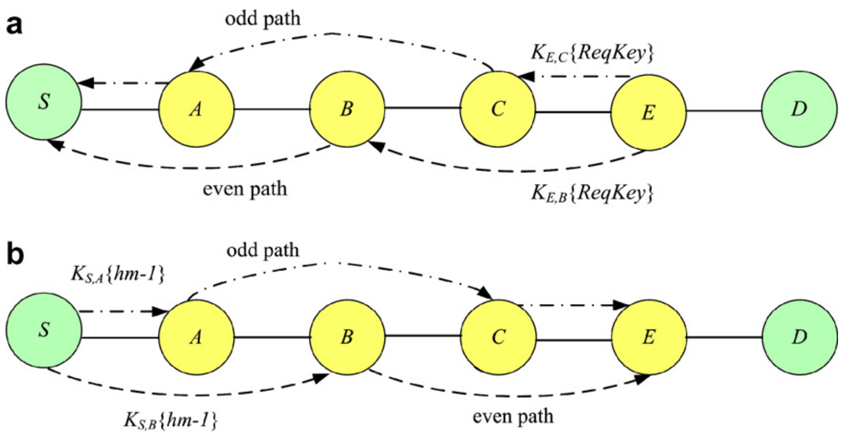

Fig. 3. An example of key disclosure procedure through the odd path (the dot dash line) and the even path (the dash line) forwarding. (a) Node E sends a $ReqKey$ to node S. (b) Node S sends h_{m-1} key to E.

key h_{m-1}, and checks to see if the computed result is equal to the received $MAC_{h_{m-1}}\{SK_i\}$. If it is match, the key segment received from node B is correct. Otherwise, node B is a malicious node. Similarly, we can identify node C in the same way. If there is only one misbehaving node, we can isolate it and exclude it from the network.

We summary our end-to-end key establishment protocol as follows:

Algorithm: End-to-End key Establishment Protocol

Assuming nodes S and D want to establish a pair-wise path key $K_{S,D}$.

Initially: Source node S generates a one-way hash chain $h_1, h_2, \ldots, h_m$. Then a node-disjoint routing protocol is used to find n node-disjoint paths from S to D. Each intermediate node on the node-disjoint paths will keep the nth hash value h_m.

For source node S:

Step 1: The source node splits the end-to-end path key $K_{S,D}$ into n key segments, SK_i, for $1 \leqq i \leqq n$.

Step 2: Assuming the ith node-disjoint path consists of nodes $S, a1, a2, \ldots, am$, and D. The source node encrypts the ith key segment SK_i with the pair-wise keys of its one-hop and two-hop neighbors denoted as $K_{S,a1}\{SK_i\}$ and $K_{S,a2}\{SK_i\}$.

Step 3: On the ith node-disjoint path, the source node will send the $K_{S,a1}\{SKi, MAC_{h_{m-1}}\{SK_i\}\}$ and $K_{S,a2}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$ messages to its next one-hop and two-hop neighbors. To achieve two-party authenticity and data integrity, we use a message authentication code (MAC). The following notations are used to denote the messages send from the source to its next one-hop and two-hop neighbors, respectively.

$$S \rightarrow a1 : K_{S,a1}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\},\ MAC_{K_{S,a1}}\{K_{S,a1}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}\}$$

$$S \rightarrow a2 : K_{S,a2}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\},\ MAC_{K_{S,a2}}\{K_{S,a2}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}\}$$

Step 4: If source node receives *ReqKey* from the odd and even paths, it encrypts the undisclosed hash key h_{m-1} with its next one-hop and next two-hop pair-wise key and forwards the encrypted message to the requested node through the reverse odd and even paths.

For each intermediate node *aj* ($1 \leqq j \leqq m$):

Step 1: If the received messages from its preceding one-hop and two-hop neighbors are consistent, it forwards the received messages to its next one-hop and two-hop neighbors. The messages forwarding are listed as follows.

$$aj \rightarrow aj+1 : K_{aj,aj+1}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\},\ MAC_{K_{aj,aj+1}}\{K_{aj,aj+1}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$$

$$aj \rightarrow aj+2 : K_{aj,aj+2}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\},\ MAC_{K_{aj,aj+2}}\{K_{aj,aj+2}\{SK_i, MAC_{h_{m-1}}\{SK_i\}\}$$

Step 2: If the received data from its preceding one-hop and two-hop neighbors are inconsistent, the intermediate node *aj* encrypts a *ReqKey* message with its preceding one-hop and two-hop neighbors' pair-wise keys, and sends the encrypted *ReqKey* to the source node via the odd path and even path.

Step 3: When the intermediate node receives the disclosed hash key h_{m-1}, it verifies whether the key is sent from the source node by checking $H(h_{m-1}) = h_m$. If the key is verified as sent from the source node, the intermediate node can identify who is a malicious node by computing the MAC of key segment with the hash key h_{m-1}.

For destination node D:

Step 1: When the destination node receives two encrypted messages from its previous one-hop and two-hop neighbors, it will check if the received messages are consistent. If both messages are consistent, node D will accept the key segment. Otherwise, it will send the *ReqKey* message to the source node to verify who is a malicious node.

Step 2: When the destination node receives enough key segments from the node-disjoint paths, it can rebuild the pair-wise path key $K_{S,D}$.

4. Security analysis and performance evaluation

Here, we analyze the security and performance of our protocol. In [12], the authors used redundant packets to prevent stop forwarding attacks happening. In our protocol, we use the (t,n) secret sharing scheme and hop by hop authentication to mitigate this attack. Hence, we can partially prevent this attack. The Byzantine attack is a malicious behavior node that alters the forwarding key to prevent the receiver from establishing a key. In our protocol, we use hop by hop authentication method to prevent Byzantine attacks. A compromised sensor node can cause not only false alarms or inject forged reports, but it can also deplete finite amounts of energy in a battery-powered network. The hop by hop authentication can also detect and filter false data. Besides, our protocol can detect compromised node as well.

To evaluate the performance of our protocol, we focus on four metrics: resilience against node capture attack, key connectivity, memory overhead, communication overhead and robustness under Byzantine attacks. We use a direct key to represent a pre-installed pair-wise key, and an indirect key to represent a pair-wise path key established via the direct key. To evaluate the resilience of our scheme against node capture attacks, we needed to know whether the fraction of additional communication links is compromised among uncompromised nodes. Assuming that when x nodes are randomly compromised in the network, let $A(x)$ denote the fraction of additional communication links that is compromised among the uncompromised nodes. In our scheme, the compromised nodes cannot gain communication links between non-compromised nodes, i.e $A(x) = 0$. Thus, the attacker cannot derive more information from the captured nodes. Hence, our proposed scheme is perfectly secure against node capture attack.

Key connectivity probability is defined as the probability in which a sensor network is securely connected. We show that our scheme can ensure that a sensor network is securely connected with high probability, as long as the network is physically connected. In our scheme, any pair of nodes in the same group have shared pair-wise keys i.e., the node can establish a secure link key. However, two nodes belonging to two neighboring groups have the probability of p_s when sharing a pair-wise key. We assume that the nodes belonging to the same group are uniformly distributed in a hexagonal area, as shown in Fig. 1. Thus, each node has about $n_r = \pi r^2 d - 1$ neighbors, where d is the node density over the sensor network, and r is the transmission radius of each node, and each group has c nodes in its zone.

We simulate key connectivity with the following assumptions. In our simulation, we assume that there are two groups and each group has 100 nodes randomly deployed in a circle with a radius = 56.4 m. Let $r = 40$ m, $d = 0.01$ and $n_r = 49$. The simulation result shows that key connectivity of our scheme is pretty high, with various p_s as shown in Fig. 4. The probability of local connectivity between neighboring groups is close to one.

The total memory overhead per sensor node in our scheme is $M = \lceil \frac{1}{2}(c-1) \rceil + \lceil \frac{1}{2}(6 \cdot c \cdot p_s) \rceil$. For example, if $p_s = 0.4$ and $c = 100$, we have $M = 170$ keys. We only measure the communication overhead of establishing an end-to-end pair-wise path key among the sensors, neglecting the communication overhead of finding node-disjoint paths. Thus, the communication overhead is the total number of packets needed in order to establish a shared key between any pair of sensors. Assuming the average path length between any pair of nodes is m ($m \geqq 4$). In our scheme, the source node needs to forward two packets to the next two intermediate nodes, the intermediate nodes on the path need to forward three packets, and the last intermediate node needs to forward two packets to the destination node. Therefore, the number of packets needed to establish a key between any pair of sensors is $n[3(m-1)-2]$, where n is the number of node-disjoint paths.

In the following, we evaluate the robustness of our hop by hop authentication scheme under a Byzantine attack. We compare our scheme to the one proposed in [12], called Ling's scheme. If an attacker does not want to get the communication content, but instead tries to paralyze the system, he can just alter the value of key segments or forge message contents. In Ling's scheme, a path is inactive if it has one compromised node on the path, excluding the source node and the destination node. However, in our scheme, a path is inactive if it two adjacent nodes on the path are compromised.

We use a simulation to compare the robustness of our scheme and Ling's scheme under a Byzantine attack. Firstly, we assume there are x random nodes captured from a network with 1000 nodes, and these captured nodes will launch Byzantine attacks. Each path has the same m intermediate nodes, and the source and destination nodes are uncompromised. Secondly, we assume that a secret key is divided into multiple shares by the secret sharing scheme, and is then delivered to the destination by multiple node-disjoint paths. We assume the path length = 20, compromised nodes x varies from 10 to 100, number of node-disjoint paths $n = 5$ and a (4, 5) secret sharing scheme is used. The simulation results are an average of 1000 times. Figs. 5 and 6 show the probability that a fraction of the paths is inactive with a different number of compromised nodes. The probability that a fraction of the paths is inactive increases quickly in Ling's scheme when the number of compromised nodes increases. But the probability that a fraction of the paths is inactive in our scheme increases very slowly when the number of compromised nodes increases. Thus, Ling's scheme is more vulnerable to Byzantine attacks than our scheme.

In our next simulation, we simulate the probability that a fraction of the paths is inactive with different path

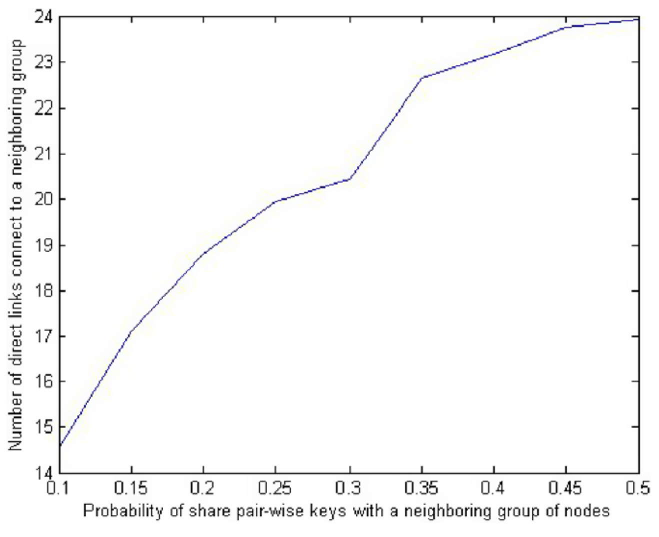

Fig. 4. Number of direct links connects to a neighboring group vs. the probability of p_s

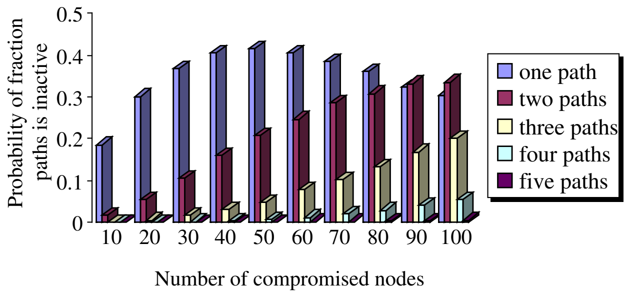

Fig. 5. The probability that a fraction of the paths is inactive vs. number of compromised nodes in Ling's scheme.

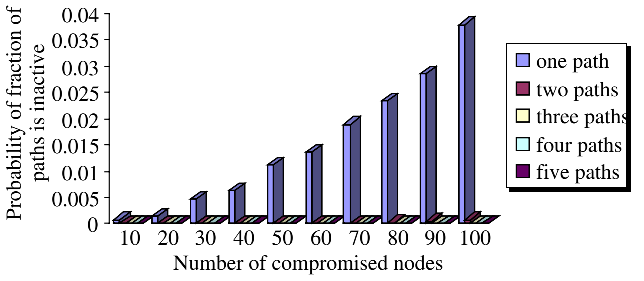

Fig. 6. The probability that a fraction of the paths is inactive vs. number of compromised nodes in our scheme.

lengths. We assume that the compromised nodes = 50, path length m varies from 10 to 50, and node-disjoint paths $n = 5$. Fig. 7 shows us that, when the path length becomes longer, the probability that a fraction of the paths is inactive rises quickly in Ling's scheme, but our scheme increases gracefully, as shown in Fig. 8. So our scheme can support a longer path length compared to Ling's scheme when establishing an end-to-end pair-wise key. Thus, our scheme can help extend network operation time when resisting malicious Byzantine attacks.

Our hop by hop authentication scheme not only can detect malicious node but also can save the energy. The energy saving is through early detection and dropping of false shared key on intermediate nodes along the forwarding path to the destination. Our hop by hop authentication scheme requires each sender to send packets to its next two hop neighbors. Since large-scale wireless sensor networks often involve very long forwarding paths, the compromised intermediate node may inject a false key value and forward it to the next hop. We show that our authentication scheme drops a false key value within two hops except when two adjacent nodes collude by sending false data intentionally. So our scheme is highly effective in filtering false key shares during their early transmission stages, thereby saving the precious energy of legitimate nodes.

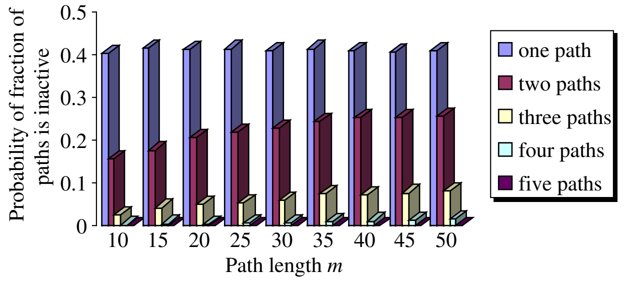

Fig. 7. The probability of fraction of paths is inactive vs. different path lengths in Ling's scheme.

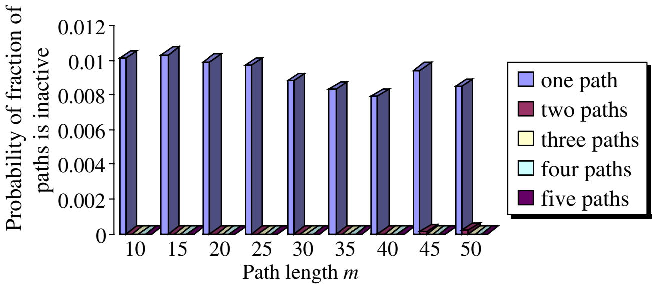

Fig. 8. The probability of fraction of paths is inactive vs. different path lengths in our scheme.

5. Conclusion

Many random key pre-distribution schemes have been developed recently to establish pair-wise keys for wireless sensor networks. But these previous schemes have a drawback in establishing a path key, which may lead to per hop key exposure problems if a node along the path is compromised. Although a number of recent research efforts have addressed this problem, most of them cannot prevent stop forwarding or Byzantine attacks. In our approach, we developed a pair-wise path key establishment scheme by taking advantage of deployment knowledge. With the help of deployment knowledge, we achieved a higher degree of connectivity with low memory requirement. We applied a hop by hop authentication scheme to improve security and robustness of pair-wise key establishment in sensor networks. Our hop by hop authentication scheme was able to detect malicious nodes and false messages by legitimate nodes. It can detect and filter false data not beyond two hops and also achieve energy savings via its early detection and its ability to drop false data. The simulation results showed that our scheme achieved more robustness in establishing a path key compared to Ling's scheme. In addition, our scheme was able to tolerate more compromised nodes and support longer path length under a Byzantine attack compared to Ling's scheme.

References

[1] L. Eschenauer, V.D. Gligor, A key-management scheme for distributed sensor networks, in: Proceedings of the 9th ACM conference on Computer and Communication Security, November 2002, pp. 41–47.

[2] H. Chan, A. Perrig, D. Song, Random key pre-distribution schemes for sensor networks, in: Proceedings of IEEE Symposium on Security and Privacy, May 2003, pp. 197–213.

[3] W. Du, J. Deng, Y.S. Han, P.K. Varshney, A pair-wise key predistribution scheme for wireless sensor networks, in: Proceedings of the 10th ACM Conference on Computer and Communication Security, October 2003, pp. 42–51.

[4] D. Liu, P. Ning, Establishing pair-wise key establishments in distributed sensor networks, in: Proceedings of 10th ACM Conference on Computer and Communications Security, October 2003, pp. 52–61.

[5] W. Du, J. Deng, Y.S. Han, S. Chen, P.K. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, in: Proceedings of IEEE INFOCOM, March 2004.

[6] D. Liu, P. Ning, Location based pair-wise key establishments for static sensor networks, in: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, 2003, pp. 72–82.

[7] D. Huang, M. Mehta, D. Medhi, H. Lein, Location aware key management scheme for wireless sensor networks, in: Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks, October 2004, pp. 29–42.

[8] Y. Zhou, Y. Zhang, Y. Fang, LLK: a link layer key establishment scheme in wireless sensor networks, in: Proceedings of IEEE Wireless Communications and Networking Conference, March 2005, pp. 29–42.

[9] Z. Yu, Y. Guan, A robust group-based key management scheme for wireless sensor networks, in: Proceedings of IEEE Wireless Communications and Networking Conference, 2005.

[10] H. Chan, A. Perrig, PIKE: peer intermediaries for key establishment in sensor network, in: Proceedings of IEEE INFOCOM, March 2005.

[11] S. Zhu, S. Xu, S. Setia, S. Jajodia, Establishing pair-wise keys for secure communication in ad hoc networks: a probabilistic approach, in: Proceedings of 11th IEEE International Conference on Network Protocols, November 2003.

[12] H. Ling, T. Znati, End to end pair-wise key establishment using multi-path in wireless sensor network, in: Proceedings of the IEEE Global Communications Conference, December 2005.

[13] G. Li, H. Ling, T. Znati, "Path key establishment using multiple secured paths in wireless sensor networks, in: Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology, 2005, pp. 43–49.

[14] L. Zhou, J. Ni, C.V. Ravishankar, Efficient key establishment for group-based wireless sensor deployments, in: Proceedings of the 4th ACM Workshop on Wireless security, September 2005, pp. 1–10.

[15] A. Shamir, How to share a secret, in: Proceedings of Communications of the ACM, vol. 22, November 1979, pp. 612–613.

[17] K. Balakrishnan, J. Deng, P.K. Varshney, TWOACK: preventing selfishness in mobile ad hoc networks, in: Proceedings of the IEEE Wireless Communications and Networking Conference, vol. 4, 2005, pp. 2137–2142.

[18] X. Li, L. Cuthbert, Node-disjointness based multi-path routing for mobile ad hoc networks, in: Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, October 2004.

[19] A. Wacker, M. Knoll, T. Heiber, K. Rothermel, A new approach for establishing pair-wise keys for securing wireless sensor networks, in: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, November 2005, pp. 27–38.

[20] R.L. Rivest, A. Shamir, L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems, in: Proceedings of Communications of the ACM, 1987, pp. 120–126.

[22] D. Huang, D. Medhi, A Byzantine resilient multi-path key establishment scheme and its robustness analysis for sensor networks, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, April 2005.

[23] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory (1976) 644–654.

[24] L. Lamport, Password authentication with insecure communication, Communications of the ACM (1981) 770–772.

[25] M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, in: Proceedings of Advances in Cryptology – Crypto'96, 1996, pp. 1–15.

Jang-Ping Sheu received the B.S. degree in computer science from Tamkang University, Taiwan, Republic of China, in 1981, and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Taiwan, Republic of China, in 1983 and 1987, respectively. He is currently a Professor of the Department of Computer Science and Information Engineering, National Central University. He was a Chair of Department of Computer Science and Information Engineering, National Central University from 1997 to 1999. He was a Director of Computer Center, National Central University from 2003 to 2006. His current research interests include wireless communications and mobile computing.

He was an associate editor of Journal of the Chinese Institute of Electrical Engineering, Journal of Information Science and Engineering, Journal of the Chinese Institute of Engineers, and Journal of Internet Technology. He is an associate editor of the IEEE Transactions on Parallel and Distributed Systems, International Journal of Ad Hoc and Ubiquitous Computing, and International Journal of Sensor Networks.

He received the Distinguished Research Awards of the National Science Council of the Republic of China in 1993–1994, 1995–1996, and 1997–1998. He received the Distinguished Engineering Professor Award of the Chinese Institute of Engineers in 2003. He received the certificate of Distinguished Professorship, National Central University in 2005. He received the K.-T. Li Research Breakthrough Award of the Institute of Information and Computing Machinery. Dr. Sheu is a senior member of the IEEE, a member of the ACM, and Phi Tau Phi Society.

Jui-Che Cheng received the B.S. degree in computer science and information engineering from Tunghai University, Taichung, Taiwan, Republic of China, in 2004 and the M.S. degree in computer science and information engineering from National Central University, Jhongli, Taiwan, Republic of China, in 2006, respectively. His current research interests include security in wireless sensor networks and mobile computing.