Future Generation Computer Systems 37 (2014) 108–116

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Secure and energy-efficient data aggregation with malicious aggregator identification in wireless sensor networks[☆]

Hongjuan Li[a], Keqiu Li[a,*], Wenyu Qu[b], Ivan Stojmenovic[c]

[a] School of Computer Science and Technology, Dalian University of Technology, Dalian, 116024, China
[b] School of Information Science and Technology, Dalian Maritime University, Dalian, 116026, China
[c] SITE, University of Ottawa, Ontario, K1N 6N5, Canada

HIGHLIGHTS

- We propose a secure and energy-efficient data aggregation scheme.
- Our scheme can detect malicious aggregators with a constant per node communication overhead.
- Theoretical analysis and extensive simulations indicate that our scheme outperforms the state-of-art secure aggregation scheme.
- We provide some extension directions to refine our scheme.

ARTICLE INFO

Article history:
Received 16 April 2012
Received in revised form
11 November 2013
Accepted 3 December 2013
Available online 24 December 2013

Keywords:
Wireless sensor networks
Data aggregation
Security
Malicious aggregator identification

ABSTRACT

Data aggregation in wireless sensor networks is employed to reduce the communication overhead and prolong the network lifetime. However, an adversary may compromise some sensor nodes, and use them to forge false values as the aggregation result. Previous secure data aggregation schemes have tackled this problem from different angles. The goal of those algorithms is to ensure that the Base Station (BS) does not accept any forged aggregation results. But none of them have tried to detect the nodes that inject into the network bogus aggregation results. Moreover, most of them usually have a communication overhead that is (at best) logarithmic per node. In this paper, we propose a secure and energy-efficient data aggregation scheme that can detect the malicious nodes with a constant per node communication overhead. In our solution, all aggregation results are signed with the private keys of the aggregators so that they cannot be altered by others. Nodes on each link additionally use their pairwise shared key for secure communications. Each node receives the aggregation results from its parent (sent by the parent of its parent) and its siblings (via its parent node), and verifies the aggregation result of the parent node. Theoretical analysis on energy consumption and communication overhead accords with our comparison based simulation study over random data aggregation trees.

1. Introduction

Wireless sensor networks (WSNs) are becoming increasingly popular to provide solutions to many security-critical applications such as wildfire tracking, military surveillance, and homeland security [1]. In sensor networks, thousands of sensor nodes collectively monitor an area. As all the sensor nodes in an area usually detect common phenomena, there is high redundancy in the raw data. To save energy and prolong network lifetime, an efficient way is to aggregate the raw data before they are transmitted to the base station as the sensor nodes are resource limited and energy constrained.

Data aggregation [2–6] is an essential paradigm to eliminate data redundancy and reduce energy consumption. During a typical data aggregation process, sensor nodes are organized into a hierarchical tree rooted at the base station. The non-leaf nodes act as aggregators, fusing data collected from their child nodes and forwarding the aggregated results towards the BS. However, data aggregation is challengeable in some applications due to the fact that the sensor nodes are vulnerable to physical tampering, which may lead to the failure of data aggregation. The sensor nodes are often deployed in hostile and unattended environments, and are not made tamper-proof due to cost considerations. So they might be captured by an adversary, which may arbitrarily tamper with the data to achieve its own purpose. Therefore, an important issue in applying data aggregation is to avoid such tampering so that the base station can get the correct data aggregation result.

☆ This work is supported by the National Natural Science Foundation of China under Grant nos. 90718030, 90818002, 60973155, and 60973117, and New Century Excellent Talents in University (NCET) of the Ministry of Education of China.

* Corresponding author.
E-mail address: keqiu@dlut.edu.cn (K. Li).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.021

To meet this challenge, some work has been done [7–12] in the area of secure data aggregation. For example, Chan et al. [7] put forward a secure hierarchical in-network aggregation scheme that provides favorable and impressive security properties. This scheme can verify whether or not tampering has occurred on the path between a leaf and the root [7]. Nevertheless, it cannot pinpoint the exact node where the tampering has happened in the case of tampering. To the best of our knowledge, none of the existing work is able to identify the nodes that tamper with the intermediate aggregation results.

To overcome this deficiency, we present a secure and energy-efficient data aggregation scheme termed MAI [13] to effectively locate the malicious aggregators in wireless sensor networks. To accomplish malicious aggregator identification, MAI performs aggregation recalculation. Since data aggregation is executed on the path from a leaf node to the base station, each node can verify its parent's aggregation by recalculating the aggregation result according to the results obtained from its siblings. If an inconsistency occurs, the parent node is flagged as a malicious node; otherwise, it is a normal one. Another characteristic of the scheme is that the aggregation and verification can be executed interactively. A parent node executes data aggregation only after the verification on its child nodes is completed. If any child node is identified to be malicious, the aggregation stops. This can avoid unnecessary wrong data transmissions and further reduce the energy consumption. Moreover, the verification procedure is a localized one, which results in a low communication overhead.

The rest of the paper is organized as follows: In Section 2, we overview some related work on secure data aggregation. Section 3 describes our system model and the attack model. In Section 4, we give a detailed description on the proposed MAI. Theoretical analysis and discussions are also presented in this section to further explain our scheme. Section 6 reports the simulation results. Finally, we summarize our work and conclude the paper in Section 7.

2. Related work

Data aggregation has the benefit to achieve bandwidth and energy efficiency. There has been extensive research [14–16] on data aggregation in various application scenarios. These aggregation schemes have been designed without security in mind. However, wireless sensor networks are likely to be deployed in hostile environments such as the battlefield, where an adversary may compromise nodes and manipulate the data.

Secure data aggregation [17,18] is a hot research problem in some applications. Basically, there are two types of aggregation models, i.e., the single-aggregator model and the multiple-aggregator model.

The authors in [8,9] investigated secure data aggregation for the single-aggregator model. The secure information aggregation (SIA) protocol presented by Przydatek et al. [8] was the first one to propose the aggregate–commit–prove framework. In this model, the BS is the only aggregator. Du et al. [9] proposed a scheme using multiple witness nodes as additional aggregators to verify the integrity of the aggregated result. As for the single-aggregator model, the corresponding schemes do not provide per-hop aggregation.

The multiple-aggregator model employs more than one aggregator. Hu and Evans [12] presented a secure aggregation protocol that is resilient to single aggregator compromise. However, this protocol cannot deal with the situation where there exist two consecutive colluding compromised aggregator nodes in the tree. Yang et al. [10] proposed SDAP, which utilizes a novel probabilistic grouping technique to dynamically subdivide an aggregation tree into subtrees of similar sizes, each of which reports its aggregation result. Suspicious groups participate in an attestation process to prove the correctness of its group aggregation. Due to the statistical nature, SDAP may not be able to detect the attacks that slightly change the intermediate aggregation results.

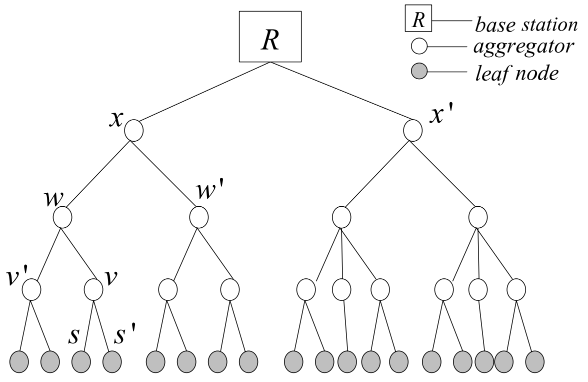

Fig. 1. An example aggregation tree.

In the privacy-preservation domain, Castelluccia et al. [19] proposed a new homomorphic encryption scheme in which the aggregation is carried out by aggregating the encrypted data at intermediate sensors without decrypting them, resulting in a higher level privacy. He et al. [20] proposed two privacy-preserving data aggregation schemes CPDA and SMART for additive aggregation functions.

3. Network model and attack model

In this section, we introduce the preliminary knowledge, including the network model and the attack model.

3.1. Network model

We model a wireless sensor network as a graph consisting of a set of n resource-limited sensor nodes $U = \{u_1, u_2, \dots, u_n\}$, each of which has an unique identifier ID_{u_i}. In addition, a resource-enhanced BS R is deployed to connect the sensor network to the outside infrastructure, e.g. the Internet. We assume that a topological tree rooted at R is constructed to perform the data aggregation. There are three types of nodes in the sensor network: leaf nodes, intermediate nodes, and the base station. The leaf nodes are collecting sensor readings. An intermediate node acts as an aggregator, aggregating the data transmitted from its child nodes and forwarding the aggregation result to its parent node. The base station is the node where the final result is aggregated. An example of such an aggregation tree is shown in Fig. 1. One method for constructing such an aggregation tree can be found in TAG [6].

Our scheme assumes that the network utilizes an identity-based public key crypto-system, which is also used in [21]. Each sensor node $u \in U$ is deployed with a private key, K_u^{-1}, and other nodes can calculate u's public key based on its ID, i.e., $K_u = f(\mathrm{ID}_u)$. Traditionally, it is assumed that public key systems exceed the memory and computational capacity of the sensor nodes. However, public key cryptography on new sensor hardware may not be as prohibitive as is traditionally assumed [21].

We further assume that the sensor nodes have the ability to perform symmetric-key encryption and decryption as well as to compute a collision-resistant cryptographic hash function. We also assume that there is a reliable transmission mechanism such that the packets transmitted in our scheme will not be lost.

3.2. Attack model

In this paper, we focus on defending against the attacks tampering with the intermediate aggregation results to make the

BS accept a false value. In many situations, aggregation results received by the BS provide a basis for critical decisions; hence, false or biased aggregation may cause catastrophic consequences. Without loss of generality, we assume that all the intermediate nodes are aggregators. We claim that our analysis can be easily extended to the case in which only some of the intermediate nodes are aggregators. The goal of our design is to localize the exact aggregator(s) that performs the malicious tampering.

In this paper, we do not consider the value changing attack where a compromised node forges a false reading on its own behalf. As indicated in [10,12], the impact of such an attack is usually limited. Besides, such a compromised node is more likely a faulty sensor node. Some other studies have targeted the identification of faulty sensors [22–24].

4. Secure and energy-efficient data aggregation with malicious aggregator identification

In this section, we present a secure and energy-efficient data aggregation with malicious aggregator identification (MAI). For simplicity, we describe our scheme for the SUM aggregation function. However, our design supports various other aggregation functions such as MAX/MIN, MEAN, COUNT, and so on. We apply our scheme on the aggregation tree shown in Fig. 1.

4.1. Aggregation commitment

Before describing the details of the proposed scheme, we first introduce the format of the packets transmitted during the aggregation. Each node has an associated packet to represent its data that is transmitted to its parent. Such a packet has the following format:

$$\langle id, count, value, signature\rangle$$

where *id* is the node's ID, *count* is the number of leaves in the subtree rooted at this node, *value* is the aggregation result computed over all the leaves in the subtree, and *signature* is a commitment computed by the node using its private key. We call the *signature* a proof. If an adversary compromises an aggregator and wants to send an invalid aggregation result, it has to forge the proof on the invalid result.

The packet for node u_i can be inductively expressed as:

$$\langle u_i, C_i, V_i, S_i\rangle$$

where $S_i = \{H(u_i\|C_i\|V_i)\}K_{u_i}^{-1}$ and $H(u_i\|C_i\|V_i)$ is a cryptographic hash function over the packet value.

If u_i is a leaf node, then $C_i = 1$ and $V_i = r_{u_i}$, where r_{u_i} is the data collected by node u_i. If u_i is an intermediate node having child nodes v_j $(j = 1, 2, \ldots, k)$ with packets $\langle v_j, C_j, V_j, S_j\rangle$, then $C_i = \sum_{j=1}^{k} C_j, V_i = \sum_{j=1}^{k} V_j$.

The pairwise key shared between u_i and its parent node is used to encrypt the packet. This encryption in practice provides not only confidentiality but also authentication. Using encryption saves the bandwidth that will otherwise be used for an additional message authentication code (MAC) [10].

Since there exist three types of nodes in the sensor network, we will respectively introduce the aggregation process executed on each type of the nodes.

(1) *Leaf node aggregation:* Data aggregation starts from the leaf nodes towards the BS. Since a leaf node does not need to do aggregation, the value in its packet is just the sensor reading. For example, in the case of Fig. 1, the leaf node *s* sends to its parent v the following packet:

$$s \to v : \langle s, 1, r_s, \{H(s\|1\|r_s)\}K_s^{-1}\rangle$$

where $\|$ denotes the stream concatenation and r_s is the sensor reading by node *s*. This packet is encrypted using the pairwise key shared between *s* and v.

(2) *Intermediate node aggregation:* When an intermediate node receives an aggregated report from one of its children, it verifies the signature of the report and keeps a copy locally (used by the aggregation verification phase) before further aggregation is performed. More specifically, an intermediate node first decrypts the report using its pairwise key shared with its child node. It then performs some simple checking on the validity of the report. The value of each item should fall in a certain range, and the verification signature should be matched with that of the report. The signature of the report is verifiable because the intermediate node can calculate the public keys of its child nodes using their IDs. If the report does not pass this check, the packet will be discarded; otherwise, the readings of all the reports received from its children will be aggregated. A new count is calculated as the sum of the count values in all the received reports. Furthermore, a new signature is calculated and attached to the end. For the example shown in Fig. 1, node w is the parent of node v. The packet that v sends to w is shown as follows:

$$v \to w\langle v, 2, Agg_v, \{H(v\|2\|Agg_v)\}K_v^{-1}\rangle$$

where "2" is the count value summed over the count values of s and s', and Agg_v is the aggregation value over r_s and $r_{s'}$. Similarly, node w sends a packet to its parent x in such a format:

$$w \to x : \langle w, 4, Agg_w, \{H(w\|4\|Agg_w)\}K_w^{-1}\rangle.$$

Each of these packets is encrypted with the pairwise key shared between the corresponding sender and its parent.

(3) *BS aggregation:* After the BS receives the aggregated data from all its children, it decrypts them, verifies their signatures and stores them locally. Then it computes the final aggregation result just like a regular intermediate node does. As such, the final aggregation result in the BS for the example shown in Fig. 1 is as follows:

$$\text{BS} : \langle R, 18, Agg_R, \{H(R\|18\|Agg_R)\}K_R^{-1}\rangle.$$

4.2. Aggregation verification

The purpose of our scheme is to enable each sensor node to independently verify whether or not its parent has done the right aggregation operation. The verification is performed by recalculating the aggregation result using its own value and the values from all its siblings, then comparing the calculated result with the one of its parent. If an inconsistency occurs at a parent node, the parent node is identified as a malicious node.

Before we present the details of our verification procedure, a high level overview of the process is introduced as follows. First, each sensor node gets the values of all its siblings (called sibling values) and the aggregation result of its parent node. Then it independently verifies whether or not its parent's aggregation result equals the recalculated one based on its own value and the received sibling values. If not, an alarm is raised (for example, using broadcast) to warn the entire network that the parent node is malicious, and the malicious node can be evicted from the network through a certain method. If no alarm is raised, all the aggregation operations are correct, and the final aggregation result can be accepted by the BS.

In what follows, we will present the detailed design of the proposed scheme.

(1) *Dissemination of the sibling packets:* To enable verification, each sensor node must get the values of its siblings in order to recalculate the aggregated value of its parent. Thus, each parent node is required to distribute the copies of the sibling packets to all

child nodes. Upon receiving the sibling packets, each node verifies their signatures, which are employed to ensure that the parent node cannot tamper with the packets of its child nodes because it does not know the private keys of its children.

(2) *Dissemination of parent packets:* To determine whether the aggregation operation is correct or not, the child nodes need to know the original aggregation result obtained by its parent node. However, a malicious parent node may tamper with the aggregation result in the aggregation phase, but send a correct result to its child nodes in the verification phase so that it can avoid being detected. In our scheme, the grandparent nodes are involved, which prevent the parent nodes from transmitting different values. Actually, it is the grandparent nodes that send the parent nodes' aggregated values to the child nodes.

As shown in the example (Fig. 1), w is the grandparent node, v is the parent node, and s is the child node. The packet w receives from v is shown as follows:

$$v \to w\langle v, 2, Agg_v, \{H(v\|2\|Agg_v)\}K_v^{-1}\rangle.$$

This packet should be sent to the child node s in the verification phase. First, w encrypts the signature of v using its own private key. In other words, the signature of w in this packet is calculated over v's signature.

$$w \to v\langle v, 2, Agg_v, \{\{H(v\|2\|Agg_v)\}K_v^{-1}\}K_w^{-1}\rangle$$

v verifies the signature and then sends the packet to s and s'.

The reason for the second signature involving two private keys is to make sure that neither the grandparent node nor the parent node can tamper with the packet, so that the packet must be the original one obtained in the aggregation phase.

(3) *Verification of the parent's aggregation:* After each sensor node gets its sibling values and its parent value, it can verify the parent's aggregation if all the packets pass the verifications on their signatures.

Each sensor node runs the same process as carried out by its parent to derive the aggregation result. This is executable as the sibling values provide all the necessary data to perform the aggregation. Once it has computed the parent aggregation result, it compares the newly derived result against the one previously received from its grandparent. If these two results are not identical, the child node immediately raises an alarm telling other nodes in the network that its parent node is malicious. Only when all the verification succeeds, the BS accepts the aggregation result.

4.3. Theoretical analysis on communication overhead

In this section, we analyze the communication overhead of our scheme and compare it with the Secure Hierarchical In-network Aggregation scheme (SHIA for short) proposed by Chan et al. [7]. SHIA is selected for comparison because it is the most related and it represents the state of the art.

Since both schemes perform similar aggregation operations, the communication overhead in the aggregation phase is also similar. Thus we only compare the communication overhead in the verification phase. To accurately measure the overhead, we use the metric *packet* $*$ *hop*, because the communication overhead is proportional to the transmission distance as each packet needs to travel several hops to arrive at the destination node. Therefore we sum up all the traveled hops for each packet as the communication overhead in the whole network.

Before we present our analytical results, we give two definitions defining the communication overhead and the off-path nodes:

Definition 1. Suppose there exist a set of packets $\{p_j | j = 1, 2 \dots z\}$ used for verification purposes. If the packet p_j needs to travel h_j hops, then the communication overhead is calculated by $\sum_{j=1}^{z} h_j$.

Definition 2. The set of off-path nodes [7] for a node u in a tree is the set of all the siblings of each of the nodes on the path from u to the root of the tree (the path is inclusive of u).

Fig. 2 shows an example of the off-path nodes for node u. The off-path nodes for node u are highlighted in bold. The path from u to the root is shaded as gray.

We assume that the aggregation tree is a complete tree with a height of h and a degree of d; hence, we have $n = \sum_{i=1}^{h} d^i$. Note that our aggregation tree is rooted at the BS, and we assume that the height of the BS is 0.

A high level overview on the verification procedure of SHIA is summarized as follows. First, the root node's aggregation results from the aggregation phase are broadcasted to every sensor node in the network. To enable the verification, each leaf node must receive all of its off-path values. Once a leaf node has received all of its off-path values, it derives the value of the root node via the computation procedure used in the aggregation phase. It is able to do so since the off-path values provide all the necessary information to perform the computation [7]. Then it compares the derived result against the broadcasted one. If they are identical, no tampering occurs on the path from the root to this leaf node; otherwise, the verification fails.

In SHIA, the communication overhead consists of two parts: the dissemination of the root value, and the dissemination of the off-path values. The root value will be sent to the entire sensor network using authenticated broadcasts, which incurs a communication overhead of n as there are n sensor nodes in the network. Hence, the communication overhead in this phase can be computed as $\sum_{i=1}^{h} d^i$.

With the knowledge of the root value, each leaf node must receive all its off-path values to enable the verification. As described in [7], the process of dissemination of the off-path values is as follows: Assume that an intermediate node t in the aggregation tree has two children u_1 and u_2. To disseminate the off-path values, t sends the packets aggregated at u_1 to u_2, and vice versa. Node t also sends any packet received from its parent to both children. See Fig. 3 for an illustration of the process. Once a node has received all the packets of its off-path nodes, it can proceed to the verification step.

In SHIA, the packets of every senor node will be sent to its sibling nodes and forwarded along the trees rooted at the sibling nodes until they reach the leaf nodes. Therefore the communication overhead can be calculated by $\sum_{i=1}^{h}(d^i \cdot \sum_{j=0}^{h-i} d^j)$.

Thus the total communication overhead needed in the verification phase of SHIA is:

$$\sum_{i=1}^{h} d^i + \sum_{i=1}^{h}\left(d^i \cdot \sum_{j=0}^{h-i} d^j\right)$$
$$= \frac{(h+1)d^{h+2} - (h+2)d^{h+1} - d^2 + 2d}{(1-d)^2}$$
$$= \Theta(n \log n).[1] \qquad (1)$$

In our scheme MAI, the communication overhead for the verification process also consists of two parts: the dissemination of the sibling values and the dissemination of the parent value.

To derive the parent aggregation result, each child node needs to get its sibling values, which indicates that each node needs to receive $(d-1)$ packets in the phase of disseminating sibling values. Since there are n nodes in the tree, the communication overhead for this step can be calculated as $n(d-1)$.

[1] h can be approximated by $\log n$.

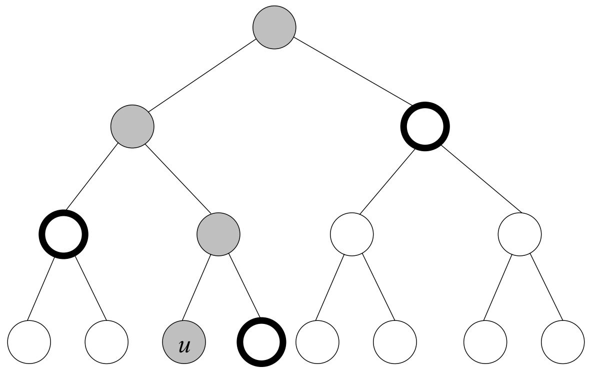

Fig. 2. The off-path nodes of u are highlighted in bold.

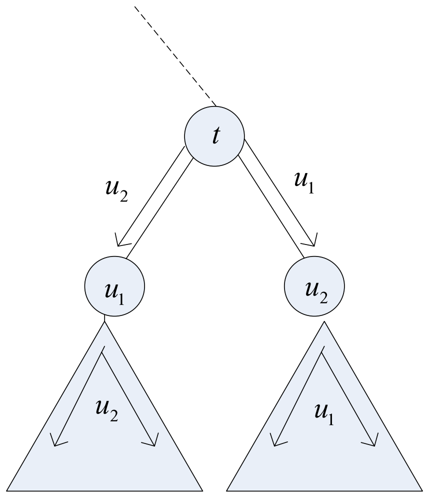

Fig. 3. Dissemination of the off-path packets.

To compare the derived aggregation result in an intermediate node with the one computed at the aggregation phase, the parent value should be disseminated to its child nodes. As every intermediate node has d children, and the parent value is sent from its grandparent, the dissemination of each parent value involves $(d + 1)$ communication overhead. Since there are $(n - d^h)$ parent nodes, the communication overhead of disseminating the parent values can be computed by $(d + 1)(n - d^h)$.

Therefore, the total communication overhead for the verification process in our MAI is calculated as follows:

$$\begin{aligned} n(d-1) + (d+1)(n-d^h) &= 2nd - d^{h+1} - d^h \\ &= \frac{d^{h+2} + d^h - 2d^2}{d-1} \\ &= \Theta(n). \end{aligned} \tag{2}$$

From Eqs. (2) and (1), we can easily see that the overhead of MAI is less than that of SHIA. This is because the verification of our scheme occurs in every parent–children connection essentially, and the reports used for verification are transmitted only within this scope, which can avoid transmitting them too far away. However, the whole network is involved for verification in the SHIA scheme. To make sure every leaf node gets its off-path values, all reports should be transmitted to the leaf nodes, which means that every report needs to travel many hops when the tree is high. This incurs extra communication overhead. Hence, our scheme is much more communicationally efficient than SHIA. Moreover, the advantage will be more obvious with the increase of the tree height.

4.4. Discussion

When a sensor network is organized into a tree topology structure, the aggregation result of every intermediate node is based on its child nodes' data. To verify that no tampering with the aggregation result at the intermediate node has occurred, let its child nodes do the same aggregation execution and then compare the result derived by the child nodes with the one computed by the parent node. In order to accurately identify the point at which forged aggregation results happen, we limit the commit-and-verify scope to every parent–children connection, and verify each intermediate aggregation result. Once there is malicious tampering at any intermediate node, we can immediately find the inconsistency between the committed aggregate and the reconstructed aggregate. In this way, we have accomplished the target of malicious node identification.

Our scheme also ensures that once an intermediate node has committed its aggregation result, which is to be verified later, all involved data must be the original data. This is because every report is sent only once from the original source and a signature is attached to each report. The signature is computed using the private key that is only known to the source, such that the report cannot be forged when it is kept at other nodes.

MAI consists of two phases: the aggregation commitment phase and the aggregation verification phase. Actually, the verification phase does not need to wait for the completion of the aggregation phase. These two phases can be executed interactively. After each grandparent node receives a packet from its child node, it may not execute the aggregation immediately. Instead, it asks its grandchild nodes to do the verification on the received packet first. Only if the verification succeeds, the grandparent node will accept the packet and do further aggregation; otherwise, the aggregation will stop. If the verification fails, it is an indication that the received packet is forged and the sending node is malicious. Such a false report, if undetected, would be forwarded to the higher level, which can cause not only the deviation in the final aggregation result but also the wastage of energy consumption. In our scheme, we detect such a false report immediately after it is sent out. In this way, we can decrease the damage of the malicious nodes and save energy.

5. Extension

5.1. Data collection at intermediate nodes

MAI assumes that only the leaf nodes collect sensor readings. Extending our scheme to support the data collection at intermediate nodes results in another problem. The aggregation result at each intermediate node will be based on the data of its child nodes and its own data. We need to get the sensor reading collected by an intermediate node to recalculate the aggregation result in the verification phase. However, the intermediate node may forge a false reading of its own in the verification phase. In this case, even if the intermediate node tempered with the aggregation result, it can escape detection by just providing its child nodes a forged reading of its own for verification, as long as the tampered aggregation result is consistent with the sum of child nodes' data and forged parent node's data. Such a node is more likely a faulty sensor, which can be detected via various existing techniques [23,24]. Therefore to extend MAI to handle the general case in which each intermediate node needs to aggregate not only the partial aggregated values from its children but also its own reading, we can employ an existing scheme to verify whether or not a node forges false data as its own reading. If not, the data aggregation and verification proceed; otherwise, the node is signaled as malicious.

In fact, by combining our scheme with the faulty sensor detection, we cannot only extend our work to more general case in which each node in the network collects data, but also prevent the value changing attack, where a compromised node forges a false reading on its own behalf. Firstly, we apply the faulty sensor detection algorithm to filter out the false sensor readings, then we deploy the malicious aggregator identification algorithm to detect the exact aggregator(s) that tempered with aggregation result(s).

For detection of faulty sensors, we provide a brief introduction of the algorithm in [24], which is effective, light and flexible with only localized information available. In this paper, the faulty sensors are also called *outliers* or *outlying sensors*.

For a dense network, each node broadcasts its sensor reading to the direct neighbors. Let $N(u)$ denote u's one-hop neighborhood. After the information collection phase, sensor u obtains the data set $E(u) = \{r_{u_i} | u_i \in N(u)\}$, formed by its neighbors' sensor readings. In our consideration, sensors should behave similarly in the close proximity. We explore the spatial correlation among the neighboring nodes to detect the outlying reports. The detection is conducted by computing the normalized distance between each reading r_{u_i} and the "center" of the data set $E(u)$.

The mean μ of $E(u)$ is considered as the "center". To compute the normalized distance, the variance of $E(u)$, denoted by σ^2 is also needed. Let $\hat{\mu}$ and $\hat{\sigma}^2$ be the estimates of μ and σ^2, respectively. A simple solution is as follows:

$$\hat{\mu} = \frac{1}{|N(u)|} \sum_{u_i \in N(u)} r_{u_i} \tag{3}$$

$$\hat{\sigma}^2 = \frac{1}{|N(u) - 1|} \sum_{u_i \in N(u)} (r_{u_i} - \hat{\mu})^2. \tag{4}$$

Such a method is simple but not reliable, since it is sensitive to the presence of outliers. Thus, the Orthogonalized Gnanadesikan–Kettenring (OGK) estimators $\hat{\mu}$ and $\hat{\sigma}^2$ [25] are employed, as described below.

Treat $E(u) = \{r_{u_i} | u_i \in N(u)\}$ as a single-variate sample set coming from a distribution with mean μ and variance σ^2. Let μ_0 and σ_0 be the median and the MAD[2] of $E(u)$, respectively. Define a weight function $W(z) = (1 - (z/c_1)^2)^2 I(|z| \le c_1)$ and a ρ-function $\rho(z) = \min(z^2, c_2^2)$, where $c_1 = 4.5$ and $c_2 = 3$. Then μ, σ^2 can be estimated respectively by [26]:

$$\dot{\mu} = \frac{\sum_{u_i \in N(u)} r_{u_i} W(m_i)}{\sum_{u_i \in N(u)} W(m_i)} \quad \text{for } m_i = \frac{r_{u_i} - \mu_0}{\sigma_0} \tag{5}$$

$$\dot{\sigma}^2 = \frac{\sigma_0^2}{|N(u)|} \sum_{u_i \in N(u)} \rho\left(\frac{r_{u_i} - \dot{\mu}}{\sigma_0}\right). \tag{6}$$

OGK provides a robust estimate for the bulk of data when there exists a small fraction of outliers in the sample data set. By standardizing the data set $E(u)$, y_i can be obtained as follows [23]:

$$y_i = \left| \frac{r_{u_i} - \dot{\mu}}{\dot{\sigma}} \right|, \quad \text{where } u_i \in N(u). \tag{7}$$

Let y_i denote the distance between each reading r_{u_i} and the "center" of the data set. If the distance is larger than a predefined threshold, the sensor u_i is identified as a faulty sensor. For the same sensor, there should be multiple neighbors reporting their decisions regarding whether or not it is outlying. The majority vote can be deployed to make the final decision.

5.2. Colluding attack

As discussed in Section 4.2, every child node needs to know its parent node's original aggregation result to determine whether the

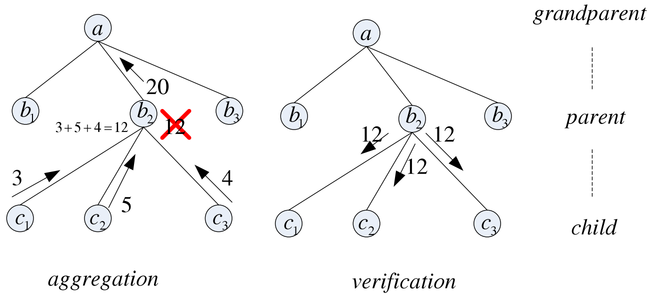

Fig. 4. A malicious parent node.

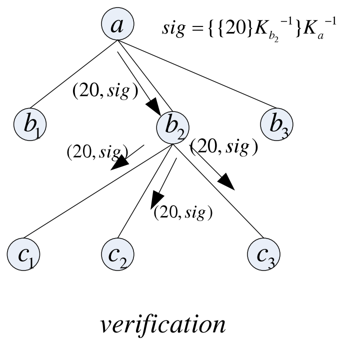

Fig. 5. Grandparent nodes are involved.

parent node tampered with it or not. However, a malicious parent node may tamper with the aggregation result in the aggregation phase and send the tempered value to the grandparent node, but send a correct value to its child nodes in the verification phase. As indicated in Fig. 4, the parent node b_2 tampers with the aggregation result 12, and sends 20 to grandparent node a in the aggregation phase. To avoid being detected by the child nodes, the parent node b_2 sends the correct value 12 to its child nodes in the verification phase. To address this problem, the grandparent nodes are involved. We let grandparent nodes send the parent nodes' values that they received in the aggregation phase to the child nodes. As indicated in Fig. 5, the parent result is sent from grandparent node a. And through attaching a signature signed by both grandparent node and parent node using their private keys, neither the grandparent nor the parent node can tamper with the packet; otherwise, there will be an inconsistency between the value and the signature.

To sum up, we let grandparent and parent nodes restrict each other when they send the parent nodes' aggregation results to the child nodes for verification. But what if they collude with each other? If they are both malicious, and they know each other's private key, the mechanism of signature will become inefficient or even useless, because they can forge a value and also a signature based on the forged value using the two private keys. As indicated in Fig. 6, parent node b_2 tampers with the aggregation result 12 and sends 20 to grandparent node a in the aggregation phase, but the child nodes get 12 for verification. The parent node can successfully escape from detection. We call this attack "colluding attack".

To address this security limitation, we propose a colluding attack detection mechanism. The grandparent node a uses 20 as the value of b_2 for aggregation. In the later verification phase, node a's aggregation result will be verified by its child nodes like b_1, b_2 and b_3. As stated in Section 4.2, the first step of aggregation verification is dissemination of sibling packets. In this case, b_2's value will be sent to b_1 and b_3. Apparently, nodes b_1 and b_3 will get 20 as b_2's value, because 20 is used in aggregation on node a. Node a surely wants all the other child nodes to use 20 for verification so that

[2] $\text{MAD}(Y) = \text{median}(|Y - \text{median}(Y)|)$.

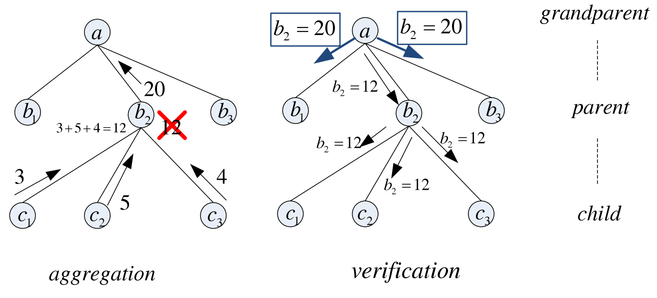

Fig. 6. Colluding attack.

there will be no inconsistency which means node a passes the verification.

It is worth noticing that, in the above scenario, child nodes c_1, c_2, and c_3 get 12 as b_2's value, while their uncle nodes b_1 and b_3 get 20 as b_2's value. We take advantage of this property to further address the colluding attack described above. Even if a child node (like c_1, c_2, and c_3) did not detect inconsistency, it needs to report to BS the parent node's value (like $b_2 = 12$). And BS also commands the child nodes' uncle nodes (like b_1 and b_3) to report their sibling nodes' (b_2) values (like $b_2 = 20$). If there is inconsistency, the colluding attack is detected.

5.3. Group keys alternative

Public key systems are assumed to be expensive in both storage and computation cost for sensor networks. To address this concern, group key mechanism can be used in this paper. For example, all the child nodes can share the same group key, and use this key for the signature and verification of the sibling nodes. The purpose of the signature in child nodes' packets is to make sure that the parent nodes cannot tamper with the child nodes' packets in the dissemination of sibling packets phase. Because the parent nodes do not know the group key of child nodes, the group key can ensure that the packets of sibling nodes used for verification are the packets used for aggregation.

In the verification of the parent packet, we use a parent private key and a grandparent private key to encrypt the signature so that neither the grandparent node nor the parent node can tamper with the packet used for aggregation. To adopt the group key, we can substitute the group key among parent and child nodes for the parent private key, and also the group key among grandparent and child nodes for the grandparent private key. Because grandparent and parent nodes do not know each other's key used to encrypt the signature, they cannot tamper with the packet, but because child nodes share the group keys with the grandparent node and parent node, they can verify the signature.

To avoid the high overhead of securely and reliably disseminating group keys from a central key server, group keys can be preloaded to the sensor nodes. Also some group key updating mechanisms [27] can be used to deal with the problem of node compromise, so that the adversary is prevented from using the captured keys.

5.4. Malicious child nodes

The verification is done by all the child nodes, however, due to Byzantine failures, such as device malfunction or attacks, the child nodes may be not reliable themselves. Some child nodes may send false reports about the decision whether the parent node is malicious or not. After we get the reports from all the child nodes, we need to do a parallel fusion and make a correct decision in spite of such Byzantine failure. This problem has been studied in [28]. And it is also our current research subject.

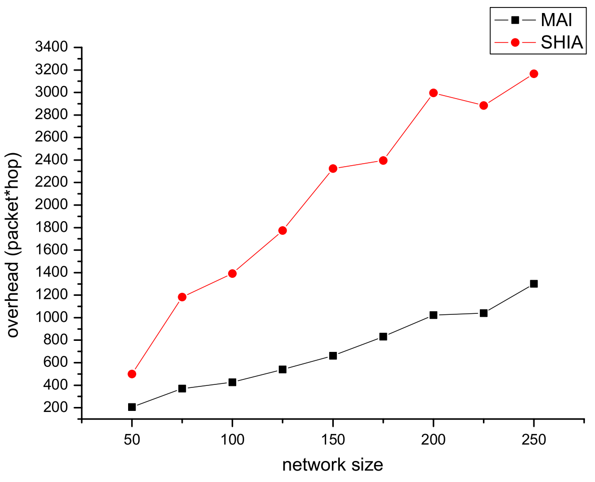

Fig. 7. Communication overhead under different network scales.

6. Simulation evaluation

The previous analytical results are applicable to a balanced tree. To evaluate the performance for more general cases, we conduct a simulation study using the NS-2 simulator to compare MAI with SHIA.

6.1. Simulation setup

In our experiments, the nodes are randomly distributed over an area. After the network is organized into an aggregation tree, we implement the two schemes on the same tree for various networking scales. The network size n varies from 50 nodes to 250 nodes. For $n < 100$, the distributed area is a 200×200 m^2 field; for $100 \leq n < 200$, the area is 300×300 m^2; and for $200 \leq n \leq 250$, it is 400×400 m^2. For each simulated topology, we adjust the communication range so that all the sensor nodes are included in the aggregation tree. In our study, we consider an energy model that sets 0.2818 W for sending or receiving a data packet per unit of time, and 100 J of total available battery power per node. The data rate is 1 Mbps. We compare the communication overhead and the energy consumption of MAI with those of SHIA and the results are reported in the following subsection.

6.2. Simulation results

Fig. 7 shows the communication overhead of MAI and that of SHIA under different network scales. We use *packet*hop* as the metric. As can be seen from Fig. 7, the overhead of MAI is much lower than that of SHIA. To further explore the dependence of the performance on the size of the aggregation tree, we report the average communication overhead per node in Fig. 8. As shown in this figure, MAI outperforms SHIA in terms of the average amount of communications. And MAI exhibits a little variance when n ranges from 50 to 250. The communication overhead is closely related to the network topology. It increases with the tree height for SHIA because the off-path values need to transmit more hops to reach the leaf nodes, and it increases with the tree degree for MAI because the child nodes need to get more sibling values to verify the parent node when the tree degree is large. In the simulations, the nodes are randomly distributed in the area; hence, the trees organized over the nodes also have different topologies for different network scales. That is why the overhead increases with the increase of the network size, but still fluctuates at some points.

Figs. 9 and 10 illustrate the energy consumption under different network scales. The percentage of the residual energy in the

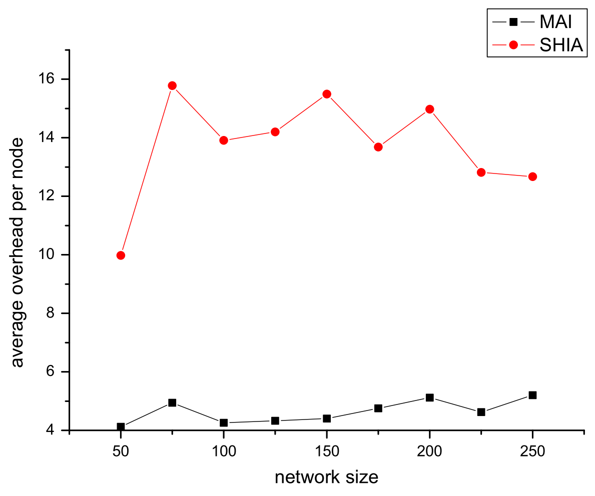

Fig. 8. Average communication overhead per node.

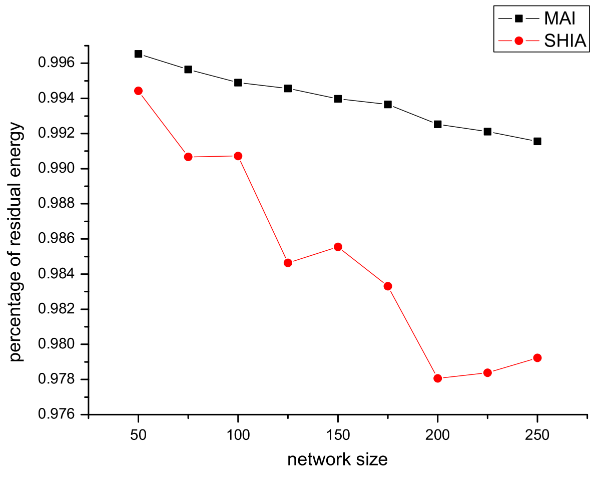

Fig. 9. Percentage of the residual energy under different network scales.

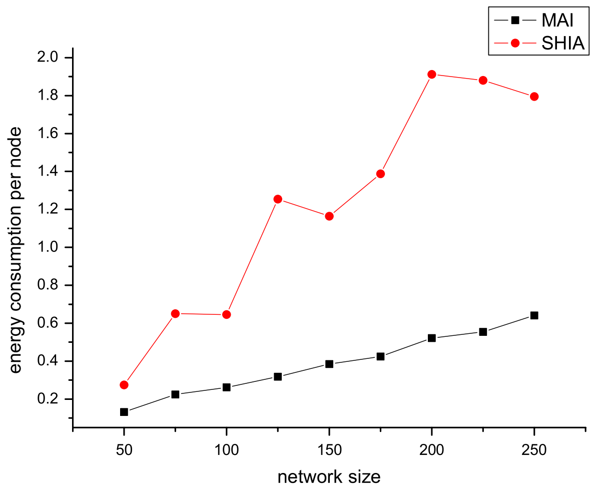

Fig. 10. Average energy consumption per node.

network with respect to the network size is shown in Fig. 9, from which we can conclude that the SHIA scheme consumes energy at a much faster pace. Fig. 10 reports the average energy consumption per node. The results indicate that our scheme is more energy efficient. This is because data transmissions contribute the major portion of the power consumption for sensor nodes, and the communication overhead of SHIA is higher than that of MAI as discussed before. Because the energy consumption is closely related to the communication overhead, our results show a general trend of increasing with increasing the network size, with some fluctuations at some points just like the results shown in Fig. 7.

In summary, the theoretical and simulation results both indicate that our proposed MAI is more efficient and effective than SHIA, as it can identify the malicious aggregators with a much lower communication overhead.

7. Conclusions

In this paper, we propose a secure and energy-efficient data aggregation scheme with malicious aggregator identification in wireless sensor networks. The goal of our proposed scheme is to make sure that not only does the BS not accept forged aggregation results, but also the malicious aggregators tampering with the intermediate results can be identified. The adversarial aggregators, after detection, can be evicted from the network, hence reducing the damage of malicious aggregators. Theoretical analysis and extensive simulations have been conducted to evaluate our scheme. The results indicate that our proposed scheme is more secure and energy efficient than SHIA, a state-of-the-art secure hierarchical in-network aggregation scheme proposed in [7]. Furthermore, we provide some extension directions to refine our scheme.

References

[1] D. Culler, D. Estrin, M. Srivastava, Overview of sensor networks, IEEE Comput. 37 (8) (2004) 41–49.
[2] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, in: Proceedings of the ACM International Conference on Mobile Computing and Networking, MobiCom, 1999, pp. 263–270.
[3] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, D. Ganesan, Building efficient wireless sensor networks with low-level naming, in: Proceedings of the ACM Symposium on Operating Systems Principles, SOSP, 2001, pp. 146–159.
[4] B. Krishnamachari, D. Estrin, S. Wicker, The impact of data aggregation in wireless sensor networks, in: Proceedings of the International Conference on Distributed Computing Systems (ICDCS) Workshops, 2002, pp. 575–578.
[5] Y. Yu, B. Krishnamachari, V.K. Prasanna, Energy-latency tradeoffs for data gathering in wireless sensor networks, in: Proceedings of the IEEE Computer and Communications Societies, INFOCOM, 2004.
[6] S. Madden, M.J. Franklin, J.M. Hellerstein, TAG: a Tiny AGgregation service for ad-hoc sensor networks, in: Proceedings of the Symposium on Operating Systems Design and Implementation, OSDI, 2002.
[7] H. Chan, A. Perrig, D. Song, Secure hierarchical in-network aggregation in sensor networks, in: Proceedings of the ACM Conference on Computer and Communication Security, CCS, 2006.
[8] B. Przydatek, D. Song, A. Perrig, SIA: secure information aggregation in sensor networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, Sensys, 2003.
[9] W. Du, J. Deng, Y. Han, P.K. Varshney, A witness-based approach for data fusion assurance in wireless sensor networks, in: Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM, 2003.
[10] Y. Yang, X. Wang, S. Zhu, G. Cao, SDAP: a secure hop-by-hop data aggregation protocol for sensor networks, in: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc, 2006.
[11] Ralph Merkle, A certified digital signature, in: Proceedings on Advances in Cryptology, 1989, pp. 218–238.
[12] L. Hu, D. Evans, Secure aggregation for wireless networks, in: Proceedings of the 2003 Symposium on Applications and the Internet Workshops, SAINTW, 2003.
[13] Hongjuan Li, Keqiu Li, Wenyu Qu, Ivan Stojmenovic, Secure and energy-efficient data aggregation with malicious aggregator identification in wireless sensor networks, in: Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing—Volume Part I, 2011, pp. 2–13.
[14] C. Itanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of the ACM International Conference on Mobile Computing and Networking, MobiCom, 2000.
[15] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, Impact of network density on data aggregation in wireless sensor networks, in: Proceedings of the International Conference on Distributed Computing Systems, ICDCS, 2002.

[16] X. Tang, J. Xu, Extending network lifetime for precision constrained data aggregation in wireless sensor networks, in: Proceedings of the IEEE Computer and Communications Societies, INFOCOM, 2006.
[17] Sankardas Roy, Mauro Conti, Sanjeev Setia, Sushil Jajodia, Secure data aggregation in wireless sensor networks, IEEE Trans. Inform. Forensics Secur. 7 (3) (2012) 1040–1052.
[18] M.K. Sandhya, K. Murugan, Secure data aggregation in wireless sensor networks using privacy homomorphism, in: Advances in Networks and Communications, 2011, pp. 482–490.
[19] C. Castelluccia, E. Mykletun, G. Tsudik, Efficient aggregation of encrypted data in wireless sensor networks, in: Proceedings of the International Conference on Mobile and Ubiquitous Systems, Mobiquitous, 2005.
[20] W. He, X. Liu, H. Nguyen, K. Nahrstedt, T. Abdelzaher, PDA: privacy-preserving data aggregation in wireless sensor networks, in: Proceedings of the IEEE Computer and Communications Societies, INFOCOM, 2007.
[21] B. Parno, A. Perrig, V. Gligor, Distributed detection of node replication attacks in sensor networks, in: Proceedings of the IEEE Symposim on Security and Privacy, SP, 2005, pp. 49–63.
[22] M. Ding, D. Chen, K. Xing, X. Cheng, Localized faulty-tolerant event boundary detection in sensor networks, in: Proceedings of the IEEE Computer and Communications Societies, INFOCOM, 2005.
[23] M. Ding, D. Chen, K. Xing, X. Cheng, Localized fault-tolerant event boundary detection in sensor networks, in: Proceedings of the IEEE Computer and Communications Societies, INFOCOM, March 13–17, 2005, pp. 902–913.
[24] F. Liu, X. Cheng, D. Chen, Insider attacker detection in wireless sensor networks, in: Proceedings of the IEEE Computer and Communications Societies, INFOCOM, May 6–12, 2007, pp. 1937–1945.
[25] D. Tyler, Robust statistics: theory and methods, J. Amer. Statist. Assoc. 103 (482) (2008) 888–889.
[26] V. Yohai, R. Zamar, High breakdown-point estimates of regression by means of the minimization of an efficient scale, J. Amer. Statist. Assoc. (1988) 406–413.
[27] W. Zhang, G. Cao, Group rekeying for filtering false data in sensor networks: a predistribution and local collaboration-based approach, in: Proceedings of the IEEE Computer and Communications Societies, INFOCOM, 2005.
[28] T. Clouqueur, K.K. Saluja, P. Ramanathan, Fault tolerance in collaborative sensor networks for target detection, IEEE Trans. Comput. 53 (3) (2004) 320–333.

Hongjuan Li received her B.E. degree in Computer Science from Dalian Jiaotong University in 2008; and her M.S. degree in Computer Science and Technology from Dalian University of Technology in 2011. She is currently pursuing her Ph.D. degree in Computer Science at Dalian University of Technology. Her research interests include cognitive radio networks, ad hoc and sensor networks, wireless and mobile security, algorithm design and analysis.

Keqiu Li is a Professor at the School of Computer Science and Engineering, Dalian University of Technology, China. He got both his bachelor's and master's degrees from Dalian University of Technology, China in 1994 and 1997, and his doctor's degree from the Japan Advanced Institute of Science and Technology in 2005. He also has five years experience in industry. Keqiu Li's research interests include Web technology, distributed computing, computer networks, grid computing, and trusted computing. He has published more than 80 technical papers in international journals and conferences, such as IEEE TPDS, ACM TOIT, and ACM TOMAPP. He is on the committee board for several international/national journals including IEEE TPDS and IEEE TC, and serves as organization chair/program chair/publication hair/program committee member for a couple of international conferences. He is a member of IEEE.

Wenyu Qu is a Professor at the School of Information and Technology, Dalian Maritime University, China. She got her bachelor's and master's degrees both from Dalian University of Technology, China in 1994 and 1997, and her doctor's degree from Japan Advanced Institute of Science and Technology in 2006. She was a lecturer at Dalian University of Technology from 1997 to 2003. Wenyu Qu's research interests include mobile-agent-based technology, distributed computing, computer networks, and grid computing. Wenyu Qu has published more than 50 technical papers in international journals and conferences. She is on the committee board for a couple of international conferences.

Ivan Stojmenovic received his Ph.D. degree in mathematics in 1985. He earned a third degree prize at the International Mathematics Olympiad for high school students in 1976. He has held regular or visiting positions in Serbia, Japan, USA, France, Spain, Brazil, Hong Kong, Taiwan, China (Distinguished Professor, Dalian University of Technology, 2010–2012), UK (Chair in Applied Computing, EECE, University of Birmingham, 2007/2008), and Canada (SITE, University of Ottawa, since 1988). He has published >250 different papers in referred journals and conferences; >100 of them are in journals with an ISI impact factor, >30 are in IEEE or ACM journals. He has co-authored over 40 book chapters. He has collaborated with >100 co-authors with Ph.D.s and a number of their graduate students from 25 different countries. He has (co)supervised >50 completed Ph.D. and master's theses, and published over 120 joint articles with supervised students. His current research interests are mainly in wireless ad hoc, sensor and robot networks. His research interests also include parallel computing, multiple-valued logic, evolutionary computing, neural networks, combinatorial algorithms, computational geometry, graph theory, computational chemistry, image processing, programming languages, and computer science education.