Group-Based Trust Management Scheme for Clustered Wireless Sensor Networks

Riaz Ahmed Shaikh, Hassan Jameel, Brian J. d'Auriol, *Member*, *IEEE Computer Society*,
Heejo Lee, *Member*, *IEEE*, Sungyoung Lee, *Member*, *IEEE*, and Young-Jae Song

Abstract—Traditional trust management schemes developed for wired and wireless ad hoc networks are not well suited for sensor networks due to their higher consumption of resources such as memory and power. In this work, we propose a new lightweight Group-based Trust Management Scheme (GTMS) for wireless sensor networks, which employs clustering. Our approach reduces the cost of trust evaluation. Also, theoretical as well as simulation results show that our scheme demands less memory, energy, and communication overheads as compared to the current state-of-the-art trust management schemes and it is more suitable for large-scale sensor networks. Furthermore, GTMS also enables us to detect and prevent malicious, selfish, and faulty nodes.

Index Terms—Trust evaluation, trust modeling, trust management, security, sensor networks.

1 Introduction

Trust in general is the level of confidence in a person or a thing. Various engineering models such as security, usability, reliability, availability, safety, and privacy models incorporate some limited aspects of trust with different meanings [1]. For example, in sensor network security, trust is a level of assurance about a key's authenticity that would be provided by some centralized trusted body to the sensor node (SN) [2], [3]. In wireless ad hoc and sensor network reliability, trust is used as a measure of node's competence in providing required service [4], [5], [6], [7]. In general, establishing trust in a network gives many benefits such as the following:

1. Trust solves the problem of providing corresponding access control based on judging the quality of SNs and their services. This problem cannot be solved through traditional security mechanisms [8].
2. Trust solves the problem of providing reliable routing paths that do not contain any malicious, selfish, or faulty node(s) [9], [10].
3. Trust makes the traditional security services more robust and reliable by ensuring that all the communicating nodes are trusted during authentication, authorization, or key management [11].

For Wireless Sensor Networks (WSNs), we visualize that trust management is a cooperative business rather than an individual task due to the use of clustering schemes such as LEACH [12], PEGASIS [13], TEEN [14], and HEED [15] in real-world scenarios. Moreover, SNs can also be deployed in the form of groups [16], which are willing to collaborate with each other in order to process, aggregate, and forward collected data [17]. This highlights the fact that these clustering schemes and group deployments enable SNs to fulfill their responsibilities in a cooperative manner rather than individually. Therefore, establishing and managing trust in a cooperative manner in clustering environment provides many advantages. Such as, within the cluster, it helps in the selection of trusted cluster head by the member nodes. Similarly, the cluster head will be able to detect faulty or malicious node(s). In case of multihop clustering [15], [18], it helps to select trusted en route nodes through which a node can send data to the cluster head. During intercluster communication, trust management helps to select trusted en route gateway nodes or other trusted cluster heads through which the sender node will forward data to the base station (BS).

A number of trust management schemes have been proposed for peer-to-peer networks [19], [20], [21] and ad hoc networks [22], [5], [23]. To the best of our knowledge, very few comprehensive trust management schemes (e.g., Reputation-based Framework for Sensor Networks (RFSN) [24], Agent-based Trust and Reputation Management (ATRM) [25], and Parameterized and Localized trUst management Scheme (PLUS) [26]) have been proposed for sensor networks. Although, there are some other works available in the literature, e.g., [6], [7], [27], [28], and so forth, that discuss trust but not in much detail. Within such comprehensive works, only ATRM [25] scheme is specifically developed for the clustered WSNs. However, this and other schemes suffer from various limitations such as these schemes do not meet the resource constraint requirements of the WSNs and, more specifically, for the large-scale WSNs. Also, these schemes suffer from higher cost associated with trust evaluation specially of distant nodes. Furthermore, existing schemes have some other limitations such as dependence on specific routing scheme, like PLUS works on the top of the PLUS_R routing scheme; dependence on specific platform, like the ATRM scheme requires

- *R.A. Shaikh, H. Jameel, B.J. d'Auriol, S. Lee, and Y.-J. Song are with the Department of Computer Engineering, Kyung Hee University, Global Campus, Seocheon-dong, Giheung-gu, Yogin-si, Gyeonggi-do, Suwon 449-701, Korea.*
  E-mail: {riaz, hassan, dauriol, sylee}@oslab.khu.ac.kr, yjsong@khu.ac.kr.
- *H. Lee is with the Division of Computer and Communication Engineering, Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, Korea.*
  E-mail: heejo@korea.ac.kr.

Manuscript received 9 Apr. 2008; revised 13 Aug. 2008; accepted 2 Dec. 2008; published online 11 Dec. 2008.
Recommended for acceptance by J.C.S. Lui.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-04-0133.
Digital Object Identifier no. 10.1109/TPDS.2008.258.

1045-9219/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society

an agent-based platform; and unrealistic assumptions, like the ATRM assumes that agents are resilient against any security threats, and so forth. Therefore, these works are not well suited for realistic WSN applications. Thus, a lightweight secure trust management scheme is needed to address these issues.

In this work, we propose a new lightweight Group-based Trust Management Scheme (GTMS) for clustered WSNs. The GTMS consists of three unique features such as

- GTMS evaluates the trust of a group of SNs in contrast to traditional trust management schemes that always focus on trust values of individual nodes. This approach gives us the benefit of requiring less memory to store trust records at each SN in the network.
- GTMS works on two topologies: intragroup topology where distributed trust management approach is used and intergroup topology where centralized trust management approach is adopted. This methodology helps to drastically reduce the cost associated with trust evaluation of distant nodes.
- GTMS not only provides a mechanism to detect malicious nodes but also provides some degree of prevention mechanism.

These and other specific features (e.g., independent of any specific routing scheme and platform and so forth) collectively make the GTMS a new, lightweight, flexible, and robust solution that can be used in any clustered WSNs.

The rest of this paper is organized as follows: Section 2 describes related work. Section 3 contains definitions, description on representation of trust value, and assumptions. Section 4 proposes trust modeling and evaluation mechanism of the GTMS. Sections 5 and 6 provide theoretical and simulation-based analysis and evaluation of the GTMS, respectively. Section 7 concludes this paper and suggests some future directions.

2 Related Work

Research work on trust management schemes for WSNs is in its infancy. To our knowledge, very few trust management schemes have been proposed such as RFSN [24], ATRM [25], and PLUS [26]. Although, there are some other works available in the literature, e.g., [6], [7], [27], [28] and so forth, that discuss trust but not in much great detail.

Ganeriwal and Srivastava [24] proposed RFSN, where each SN maintains the reputation for neighboring nodes only. Trust values are calculated on the basis of that reputation and they use Bayesian formulation for representing reputation of a node. RFSN assumes that the node has enough interactions with the neighbors so that the reputation (beta distribution) can reach a stationary state. However, if the rate of node mobility is higher, reputation information will not stabilize. In RFSN, no node is allowed to disseminate bad reputation information. If it is assumed that "bad" reputation is implicitly included by not giving out good reputation, then in that case, the scheme will not be able to cope with uncertain situations [28].

Boukerche et al. [25] have proposed an ATRM scheme for WSNs. ATRM is based on a clustered WSN and calculates trust in a fully distributed manner. ATRM works on specific agent-based platform. Also, it assumes that there is a single trusted authority, which is responsible for generating and launching mobile agents, which makes it vulnerable against a single point of failure. ATRM also assumes that mobile agents are resilient against malicious nodes that try to steal or modify information carried by the agent. In many applications, this assumption may not be realistic.

Yao et al. [26] have proposed PLUS for sensor network security. The authors adopt a localized distributed approach and trust is calculated based on either direct or indirect observations. This scheme works on top of their own defined routing scheme called PLUS_R. In this scheme, the authors assume that all the important control packets generated by the BS must contain a hashed sequence number (HSN). Inclusion of HSN in control packets not only increases the size of packets resulting in higher consumption of transmission and reception power but also increases the computational cost at the SNs. Also, whenever a judge node receives a packet from another node i, it will always check the integrity of the packet. If the integrity check fails, then the trust value of node i will be decreased irrespective of whether node i was really involved in maliciously making some modification in a packet or not. So, node i may get unfair penalty.

Recently, Liu et al. [27] have proposed a very simple trust management scheme for Resilient Geographic Routing (T-RGR). Their trust algorithm works in a fully distributed manner, in which each node monitors the behavior of one-hop neighbors. In the T-RGR scheme, authors have used many predefined threshold values that make their scheme nonadaptive. Also, in their scheme, each node only relies on its direct monitoring for calculating trust value, which makes it vulnerable against collaborative attacks.

3 Definitions, Representation, and Assumptions

3.1 Definitions

Our proposed GTMS calculates the trust value based on direct or indirect observations. Direct observations represent the number of successful and unsuccessful interactions and indirect observations represent the recommendations of trusted peers about a specific node. Here, interaction means the cooperation of two nodes. For example, a sender will consider an interaction as successful if the sender receives an assurance that the packet is successfully received by the neighbor node and that node has forwarded the packet toward the destination in an unaltered fashion. Thus

- The first requirement, i.e., successful reception, is achieved on reception of the link layer acknowledgment (ACK). IEEE 802.11 is a standard link layer protocol, which keeps packets in its cache until the sender receives an ACK. Whenever the receiver node successfully received the packet, it will send back an ACK to the sender. If the sender node did not receive the ACK during a predefined threshold time, then it will retransmit that packet.
- The second requirement, i.e., forwarding of the packet, is achieved by using enhanced passive

acknowledgment (PACK) by overhearing the transmission of a next hop on the route, since they are within the radio range [10].

If the sender node does not overhear the retransmission of the packet within a threshold time from its neighboring node or the overheard packet is found to be illegally fabricated (by comparing the payload that is attached to the packet), then the sender node will consider that interaction as an unsuccessful one. If the number of unsuccessful interactions increases, the sender node decreases the trust value of that neighboring node and may consider it as a faulty or malicious node.

3.2 Representation of Trust Value

Generally, a trust value is considered to be a numerical quantity lying between 0 and 1 (inclusive) as suggested earlier in [5], [22], and [29] or between -1 and 1 (inclusive) as described in [4] on a real number line. In this paper, we use trust value as an integer in the interval between 0 and 100 (inclusive). However, other ranges, for example base 2 ranges, could be used as well. Although presenting the trust values as a real number or integer may not play an important role in traditional networks, but for SNs this issue is of critical importance due to limited memory, and transmission, reception power. This change will give us benefits such as: Representation of trust value [0, 100] as an unsigned integer (1 byte) saves 75 percent of memory space as compared to trust values represented as a real number (4 bytes). Less number of bits need to be transmitted during the exchange of trust values between SNs. This gives us the benefit of less consumption of transmission and reception power.

3.3 Assumptions

We assume that the sensor network consists of large number of SNs that are deployed in an open or hostile environment. We also assume that all SNs have unique identities as it is also assumed in [24], [25], and [30]. In some of the sensor network models, nodes do not have unique identities like IP in traditional networks. However, in order to uniquely identify the SNs and perform communication in those environments, class-based addressing scheme [31], [32], [33] is used, in which a node is identified by a triplet <location, node type, node subtype>. We also, assume that SNs are organized into clusters with the help of any proposed clustering scheme such as [12] and [14]. We also assume that the BS is a central command authority. It has no resource constraint problem, and furthermore, it cannot be compromised by an attacker. In order to provide protection of trust values from traffic analysis or fabrication during transfer from one node to another, we assume a secure communication channel, which can be established with the help of any key management scheme [34], [35], [36], [37].

4 Group-Based Trust Management Scheme

The proposed trust model works with two topologies. One is the intragroup topology where distributed trust management is used. The other is intergroup topology where centralized trust management approach is employed. For the intragroup network, each sensor that is a member of the group calculates individual trust values for all group members. Based on the trust values, a node assigns one of the three possible states: 1) trusted, 2) untrusted, or 3) uncertain to other member nodes. This three-state solution is chosen for mathematical simplicity and is found to provide appropriate granularity to cover the situation. After that, each node forwards the trust state of all the group member nodes to the CH. Then, centralized trust management takes over. Based on the trust states of all group members, a CH detects the malicious node(s) and forwards a report to the BS. On request, each CH also sends trust values of other CHs to the BS. Once this information reaches the BS, it assigns one of the three possible states to the whole group. On request, the BS will forward the current state of a specific group to the CHs.

Our group-based trust model works in three phases: 1) Trust calculation at the node level, 2) trust calculation at the cluster-head level, and 3) trust calculation at the BS level.

4.1 Trust Calculation at the Node Level

At the node level, a trust value is calculated using either time-based past interaction or peer recommendations. Whenever a node y wants to communicate with node x, it first checks whether y has any past experience of communication with x during a specific time interval or not. If yes, then node x makes a decision based on past interaction experience, and if not, then node x moves for the peer recommendation method.

4.1.1 *Time-Based Past Interaction Evaluation*

Trust calculation at each node measures the confidence in node reliability. Here, the network traffic conditions such as congestion, delay, and so forth should not affect the trust attached to a node; this means that the trust calculation should not emphasize the timing information of each interaction too rigidly. Therefore, we introduce a sliding time window concept, which takes relative time into consideration and reduces the effects of network conditions on overall trust calculation. If real-time communication is a requirement, as is the case in most real-world applications, this timing window concept does not provide any hindrance when it comes to real-time delivery of packets. The communication protocol in such applications is always accompanied with time stamps, and thus any node that delays the delivery of packets by taking advantage of the sliding timing window will be detected straightforwardly.

The timing window (Δt) is used to measure the number of successful and unsuccessful interactions. It consists of several time units. The interactions that occur in each time unit within the timing window are recorded. After a unit of time elapses, the window slides one time unit to the right, thereby dropping the interactions done during the first unit. Thus, as time progresses, the window forgets the experiences of one unit but adds the experiences of the newer time unit. The window length could be made shorter or longer based on network analysis scenarios. A sample scenario of the GTMS time window scheme is illustrated in Fig. 1. The time window Δt consists of five units. During the first unit of Δt_1, the number of successful and unsuccessful interactions is 4 and 2, respectively, and during the whole Δt_1 interval, the number of successful and unsuccessful interactions is 29 and 15, respectively. After the passage of the first unit, the new time interval Δt_2 drops the interaction values that took place during the very first unit of Δt_1 $(S = 4, U = 2)$ and only consider the values of the last four units of Δt_1 plus values of one recent unit added on the right $(S = 6, U = 2)$.

With this time window information, the time-based past interaction trust value $(T_{x,y})$ of node y at node x that lies between 0 and 100 is defined as

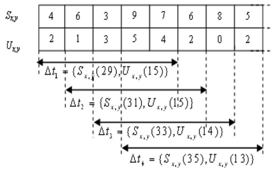

$S_{x,y}$
4 6 3 9 7 6 8 5
$U_{x,y}$
2 1 3 5 4 2 0 2
$\Delta t_1 = \{S_{x,y}(29), U_{x,y}(15)\}$
$\Delta t_2 = \{S_{x,y}(31), U_{x,y}(15)\}$
$\Delta t_3 = \{S_{x,y}(33), U_{x,y}(14)\}$
$\Delta t_4 = \{S_{x,y}(35), U_{x,y}(13)\}$

Fig. 1. Sliding time window scheme of GTMS.

$$T_{x,y} = \left[100\left(\frac{S_{x,y}}{S_{x,y}+U_{x,y}}\right)\left(1-\frac{1}{S_{x,y}+1}\right)\right] = \left[\frac{100(S_{x,y})^2}{(S_{x,y}+U_{x,y})(S_{x,y}+1)}\right], \tag{1}$$

where $[\cdot]$ is the nearest integer function, $S_{x,y}$ is the total number of successful interactions of node x with y during time Δt, $U_{x,y}$ is the total number of unsuccessful interactions of node x with y during time Δt. The expression $(1-\frac{1}{S_{x,y}+1})$ in (1) approaches 1 rapidly with an increase in the number of successful interactions. We choose this function instead of a linear function since such a function would approach very slowly to 1 with the increase in successful interactions; hence, it would take considerably longer time for a node to increase its trust value for another node. In order to balance this increase in the trust value with the increasing number of unsuccessful interactions, we multiply the expression with the factor $(\frac{S_{x,y}}{S_{x,y}+U_{x,y}})$, which indicates the percentage of successful interactions among the total interactions. Thus, this equation has a built-in capability of diminishing the effects of a few wrong declarations of interactions that may be caused by any network traffic problems.

Fig. 2 shows the behavior of time-based past interactions trust values against successful and unsuccessful interactions. When we do not get even a single successful interaction, the trust value remains 0. With an increase in successful interactions, the trust value increases but stays humble if the number of unsuccessful interactions is also considerably high. For example, with 60 unsuccessful and 50 successful interactions, the trust value is 45.

After calculating the trust value, a node will quantize trust into three states as follows:

$$Mp(T_{x,y}) = \left\{\begin{array}{ll} \text{trusted} & 100-f \le T_{x,y} \le 100 \\ \text{uncertain} & 50-g \le T_{x,y} < 100-f \\ \text{untrusted} & 0 \le T_{x,y} < 50-g \end{array}\right\}, \tag{2}$$

where f represents half of the average values of all trusted nodes, and g represents one third of the average values of all untrusted nodes. The usage of half and one third of average values in evaluation directly affects the resiliency of a node, which is discussed in Section 5.1. Both f and g are calculated as follows:

$$f_{j+1} = \begin{cases} \left[\frac{1}{2}\left(\frac{\sum_{i\in R_x} T_{x,i}}{|R_x|}\right)\right] & 0 < |R_x| \le n-1, \\ f_j & |R_x| = 0, \end{cases} \tag{3}$$

$$g_{j+1} = \begin{cases} \left[\frac{1}{3}\left(\frac{\sum_{i\in M_x} T_{x,i}}{|M_x|}\right)\right] & 0 < |M_x| \le n-1, \\ g_j & |M_x| = 0, \end{cases} \tag{4}$$

where $[\cdot]$ is the nearest integer function, R_x represents the set of trustful nodes for node x, M_x represents the set of untrustful nodes for node x, and n is the total number of nodes that contains trustful, untrustful, and uncertain nodes. At start-up, the trust values of all nodes are 50, which is an uncertain state. Initially, f and g are equal to 25 and 17, respectively, although other values could also be used by keeping the following constraint intact: $f_i - g_i \ge 1$, which is necessary for keeping the uncertain zone between trusted and untrusted zones. The values of f and g are adaptive. During the steady-state operation, these values can change with every passing unit of time, which creates dynamic trust boundaries as shown in Fig. 3. At any stage, when $|R_x|$ or $|M_x|$ becomes zero, then the value of f_{j+1} or g_{j+1} remains the same as the previous values (f_j and g_j). The nodes whose values are above $100-f$ will be declared as trustful nodes (2), and nodes whose values are lower than $50-g$ will be considered as untrusted nodes (2). After each passage of time, Δt, nodes will recalculate the values of f and g. This trust calculation procedure will continue in this fashion.

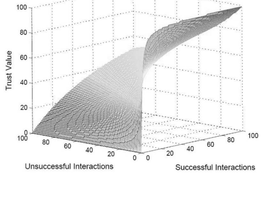


Fig. 2. Time-based past interaction evaluation.

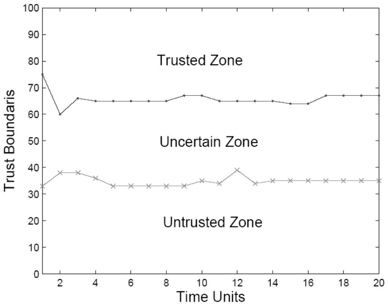


Fig. 3. Adaptive trust boundaries creation.

4.1.2 Peer Recommendation Evaluation

Let a group be composed of n uniquely identified nodes. Furthermore, each node maintains a trust value for all other nodes. Whenever a node requires peer recommendation, it will send a request to all member nodes except for the untrusted ones. Let us assume that j nodes are trusted or uncertain in a group. Then, node x calculates the trust value of node y as follows:

$$T_{x,y} = \left[\frac{\sum_{i\in \mathrm{R}_x \cup \mathrm{C}_x} T_{x,i} * T_{i,y}}{100 * j}\right]; j = |\mathrm{R}_x \cup \mathrm{C}_x| \le n-2, \tag{5}$$

where $[\cdot]$ is the nearest integer function, $T_{x,i}$ is the trust value of the recommender, and $T_{i,y}$ is the trust value of node y sent by node i. Here, $T_{x,i}$ is acting as a weighted value of the recommender that is multiplied with the trust value $T_{i,y}$, sent by the recommender, such that the trust value of node y should not increase beyond the trust value between node x and the recommender node i.

4.2 Trust Calculation at the Cluster-Head Level

Here, we assume that the CH is the SN that has higher computational power and memory as compared to other SNs.

4.2.1 *Trust State Calculation of Own Group*

In order to calculate the global trust value of nodes in a group, CH asks the nodes for their trust states of other members in the group. We use the trust states instead of the exact trust values due to two reasons. First, the communication overhead would be less as only a simple state is to be forwarded to the CH. Second, the trust boundaries of an individual node vary from other nodes. A particular trust value might be in a trusted zone for one node, whereas it may only correspond to the uncertain zone for another node. Hence, the calculation of the global trust state of nodes in a group would be more feasible and efficient if we only calculate it using the trust states.

Let us suppose there are $n+1$ nodes in the group including the CH. The CH will periodically broadcast the request packet within the group. In response, all group member nodes forward their trust states, s, of other member nodes to the CH. The variable, s, can take three possible states: trusted, uncertain, and untrusted. The CH will maintain these trust states in a matrix form, as shown below:

$$TM_{ch} = \begin{bmatrix} s_{ch,1} & s_{1,ch} & \cdots & s_{n,1} \\ s_{ch,2} & s_{1,2} & \cdots & s_{n,2} \\ \vdots & \vdots & \vdots & \vdots \\ s_{ch,n} & s_{1,n} & \cdots & s_{n,n-1} \end{bmatrix},$$

where TM_{ch} represents the trust state matrix of cluster head ch, and $s_{ch,1}$ represents the state of node 1 at cluster head ch. The CH assigns a global trust state to a node based on the relative difference in trust states for that node. We emulate this relative difference through a standard normal distribution. Therefore, the CH will define a random variable X such that

$$X(s_{i,j}) = \begin{cases} 2 & \text{when} \quad s_{i,j} = \text{trusted}, \\ 1 & \text{when} \quad s_{i,j} = \text{uncertain}, \\ 0 & \text{when} \quad s_{i,j} = \text{untrusted}. \end{cases} \tag{6}$$

Assuming this to be a uniform random variable, we define the sum of m such random variables as S_m. The behavior of S_m will be that of a normal variable due to the central limit theorem [38]. The expected value of this random variable is m and the standard deviation is $\sqrt{m/3}$. The CH defines the following standard normal random variable for a node j:

$$Z_j = \frac{\sqrt{3}\Big(X(s_{ch,j}) + \sum_{i=1, i\neq j}^{m} X(s_{i,j}) - m\Big)}{\sqrt{m}}. \tag{7}$$

If $Z_j \in [-1, 1]$, then node j is termed as uncertain, else if $Z_j > 1$, it is called trusted. If $Z_j < -1$, it is labeled as untrusted.

4.2.2 *Trust Calculation of Other Groups*

During group-to-group communication, the CH maintains the record of past interactions of another group in the same manner as individual nodes keep record of other nodes. Trust values of a group are calculated on the basis of either past interaction or information passed on by the BS. Here, we are not considering peer recommendations from other groups in order to save communication cost. Let us suppose CH i wants to calculate the trust value ($T_{i,j}$) of another cluster j. Then, it can be calculated by using either time-based past interaction ($PI_{i,j}$) evaluation or by getting recommendation from the BS ($BR_{i,j}$) as shown below:

$$T_{i,j} = \left\{ \begin{array}{ll} \left[\frac{100(S_{i,j})^2}{(S_{i,j}+U_{i,j})(S_{i,j}+1)}\right] & \text{if } PI_{i,j} \neq \varphi \\ BR_{i,j} & \text{if } PI_{i,j} = \varphi \end{array} \right\}. \tag{8}$$

If the cluster head does not have any record of past interactions within the time window, i.e., $PI_{i,j} = \varphi$, it requests the BS for the trust value.

4.3 Trust Calculation at Base Station Level

The BS also maintains the record of past interactions with CHs in the same manner as individual nodes do, as shown below:

$$T_{BS,ch_i} = \left[\frac{100\,(S_{BS,ch_i})^2}{\big(S_{BS,ch_i} + U_{BS,ch_i}\big)\big(S_{BS,ch_i} + 1\big)}\right], \tag{9}$$

where $[\cdot]$ is the nearest integer function, $S_{BS,ch}$ is the total number of successful interactions of BS with CH during time Δt, and $U_{BS,ch}$ is the total number of unsuccessful interactions of BS with CH during time Δt.

Let us suppose there are $|G|$ groups in the network. BS periodically multicasts request packets to the CHs. On request, the CHs forward their trust vectors, related to the recommendations of other groups based upon past interactions, to BS as given by $\overrightarrow{T}_{ch} = (T_{ch,1}, T_{ch,2}, \ldots, T_{ch,|G|-1})$.

On reception of trust vectors from all the CHs, the BS will calculate the trust value of each group in a manner shown below:

$$\begin{aligned} T_{BS,G_1} &= \left[\frac{\sum_{i=1}^{|G|-1} \big(T_{BS,ch_i}\big)\big(T_{G_i,G_1}\big)}{|G|-1}\right], \ldots, \\ T_{BS,G_m} &= \left[\frac{\sum_{i=1}^{|G|-1} \big(T_{BS,ch_i}\big)\Big(T_{G_i,G_{|G|}}\Big)}{|G|-1}\right], \end{aligned} \tag{10}$$

where $T_{BS,ch}$ is the trust value of the CH i at the BS, $T_{Gi,G1}$ is the trust value of group G_1 at group G_i, and $|G|$ represents the total number of groups in the network.

5 Theoretical Analysis and Evaluation

5.1 Security Resilience Analysis

In this section, we analyze the resiliency of the GTMS protocol against attacks on trust management. We broadly categorize two types of nodes: good ones and bad ones. Our assumption is that good nodes interact successfully most of the time and submit true recommendations. On the other hand, bad nodes try to do as many unsuccessful interactions as possible and send false recommendations about good nodes. Clearly, this concept of good and bad nodes is relative. A node might be a good node in the view of one node, whereas it might be bad for another. In the following, we define this concept more rigorously, capture the behavior of bad nodes, and model how they might try to get unfair advantage in our trust model. Then, we prove our protocol's resilience against such bad behaviors. This analysis can be applied straightaway to higher level groups in a modular way.

We begin with the notion of bad behavior and unfair advantage. Both of these attributes define a malicious node. The goal of a malicious node while interacting with other nodes is to do as many unsuccessful interactions as possible while keeping the following objectives intact:

- obtain a higher trust value for itself than the actual calculated trust value; more importantly, to get into the trusted zone when its rightful place is in the uncertain or untrusted zone,
- decrease the trust value of a good node if possible, and
- increase the trust value of a collaborating bad node if possible.

After defining a malicious node's objectives in this way, we can prove that our trust management scheme at the node level is resilient against malicious nodes if it can stop the malicious nodes from fulfilling their objectives. Apparently, it is hard to come up with a scheme that can totally stop such behavior. However, if we can quantify the limits of such nodes, we can have a certain amount of assurance for our system. This assurance ensures that a *smart* node, which tries to minimize the number of successful interactions with other nodes while still being in the trusted zone, cannot accomplish its goals but within certain limits. More precisely, the *smart* node has to maintain the number of successful interactions higher or equal to the number of unsuccessful interactions, as will be explained in the following.

5.1.1 Resilience Analysis at Node Level

In this section, we test the resilience of our trust model against malicious nodes. In what follows, we describe the interaction between nodes within a generic group G in the sensor network. Let R_i, C_i, and M_i denote the set of trusted, uncertain, and untrusted nodes for a node i. We begin with a definition of a malicious node.

Definition 5.1. *An SN m is said to be bad for a node i if it has interacted with i at least once and $U_{j,m} \geq S_{j,m}$.*

Definition 5.2. *A bad node m for a node i is said to have deceived i if $s_{i,m} = trusted$.*

Definition 5.3. *A Trust Management Scheme is said to be resilient against deception by a bad node at the node level if no bad node can deceive another node.*

Claim 1. *Our GTMS is resilient against deception by a bad node at the node level.*

Proof. Suppose to the contrary that there exists a bad node m for a node i that successfully deceived i. Then, according to the definition: $U_{i,m} \geq S_{i,m}$ and $s_{i,m} = trusted$. There are three cases.

Case 1. $S_{i,m} \geq 1$. This means that node m has interacted with node i within the time window Δt. Let a denote the real number $U_{i,m}/S_{i,m}$. So, $a \geq 1$. Now, since $s_{i,m} = trusted$, therefore at the time of the last interaction, the trust calculation was done using the past interaction evaluation. Assume first that $R_i \neq \varphi$. Then,

$$100 - \frac{\sum_{k \in \mathrm{R}_i} T_{i,k}}{2|\mathrm{R}_i|} < T_{i,m}.$$

Since i has previously interacted with node m within the time window in the past, we have

$$\begin{aligned} T_{i,m} &= 100\left(\frac{S_{i,m}}{S_{i,m}+U_{i,m}}\right)\left(1-\frac{1}{S_{i,m}+1}\right) \\ &= \frac{100}{a+1} - \frac{100}{(a+1)(S_{i,m}+1)}. \end{aligned}$$

This implies that

$$\begin{aligned} 100 - \frac{\sum_{k \in \mathrm{R}_i} T_{i,k}}{2|\mathrm{R}_i|} &< \frac{100}{a+1} - \frac{100}{(a+1)(S_{i,m}+1)} \\ \Rightarrow 100\left(1 - \frac{1}{a+1} + \frac{1}{(a+1)(S_{i,m}+1)}\right) \\ &< \frac{\sum_{k \in \mathrm{R}_i} T_{i,k}}{2|\mathrm{R}_i|} \leq \frac{100|\mathrm{R}_i|}{2|\mathrm{R}_i|}. \end{aligned}$$

The last inequality is true since all the $T_{i,k}$'s are within the trusted zone. We obtain

$$\frac{1}{2} < \frac{1}{a+1} - \frac{1}{(a+1)(S_{i,m}+1)}.$$

Since $a \geq 1$, this gives us: $\frac{1}{(S_{i,m}+1)} < 0$, which is obviously impossible. If $R_i = \varphi$, then we have

$$75 < T_{i,m} = \frac{100}{a+1} - \frac{100}{(a+1)(S_{i,m}+1)},$$

which again leads to the contradiction: $\frac{1}{(S_{i,m}+1)} < 0$.

Case 2. $S_{i,m} = 0$. We now consider $U_{i,m} \geq 1$. Let t_1 denote the first of these unsuccessful interactions within the time window Δt. For the second interaction request within the time window Δt, i must have calculated the trust value for m as

$$\begin{aligned} T_{i,m} &= 100\left(\frac{S_{i,m}}{S_{i,m}+U_{i,m}}\right)\left(1-\frac{1}{S_{i,m}+1}\right) \\ &= 100\left(\frac{0}{0+1}\right)\left(1-\frac{1}{0+1}\right) = 0. \end{aligned}$$

However, this is a contradiction, since the lower bound for the trusted zone is always higher than 0. This proves the claim.

Case 3. $S_{i,m} = 0$, $U_{i,m} = 0$. This means that node m has no interaction with node i at all within the time window Δt. In that case, node m will rely on the recommendation of trusted peers. □

Definition 5.4. *An SN m is said to be really bad for a node i if it has interacted with i at least once and $U_{i,m} \geq 2S_{i,m}$.*

Definition 5.5. *A really bad node m for a node i is said to have deceived i if $s_{j,m} = uncertain$.*

Definition 5.6. *A Trust Management Scheme is said to be resilient against deception by a really bad node at the node level if no really bad node can deceive another node.*

Claim 2. *Our GTMS is resilient against deception by a really bad node at the node level.*

Proof. Suppose to the contrary that there exists a really bad node m for a node i that deceived i. Then, according to the definition: $U_{i,m} \geq 2S_{i,m}$ and $s_{i,m} = uncertain$. We consider the three separate cases.

Case 1. $S_{i,m} \geq 1$. This means that node m has interacted with node i within the time window Δt. Let a denote the real number $U_{i,m}/2S_{i,m}$. So, $a \geq 1$. Now, since $s_{i,m} = uncertain$, therefore at the time of the last interaction, the trust calculation was done using the past interaction evaluation. First, assume that $M_i \neq \varphi$, then

$$50 - \frac{\sum_{k \in \mathrm{M}_i} T_{i,k}}{3|\mathrm{M}_i|} < T_{i,m}.$$

Since i has previously interacted with node m within the time window in the past, we have

$$\begin{aligned} T_{i,m} &= 100\left(\frac{S_{i,m}}{S_{i,m} + U_{i,m}}\right)\left(1 - \frac{1}{S_{i,m}+1}\right) \\ &= \frac{100}{2a+1} - \frac{100}{(2a+1)(S_{i,m}+1)}. \end{aligned}$$

This implies that

$$\begin{aligned} 50 - \frac{\sum_{k \in \mathrm{M}_i} T_{i,k}}{3|\mathrm{M}_i|} &< \frac{100}{2a+1} - \frac{100}{(2a+1)(S_{i,m}+1)} \\ \Rightarrow 50\left(1 - \frac{2}{2a+1} + \frac{2}{(2a+1)(S_{i,m}+1)}\right) \\ &< \frac{\sum_{k \in \mathrm{M}_i} T_{i,k}}{3|\mathrm{M}_i|} \leq \frac{50|\mathrm{M}_i|}{3|\mathrm{M}_i|}. \end{aligned}$$

The last inequality is true since all the $T_{i,k}$s are within the untrusted zone. We obtain

$$\frac{1}{3} < \frac{1}{2a+1} - \frac{1}{(2a+1)(S_{i,m}+1)}.$$

Since $a \geq 1$, this gives us: $\frac{1}{(S_{i,m}+1)} < 0$, which is again impossible. If $M_i = \varphi$, then we have

$$\frac{100}{3} < T_{i,m} = \frac{100}{a+1} - \frac{100}{(a+1)(S_{i,m}+1)},$$

which again leads to the contradiction: $\frac{1}{(S_{i,m}+1)} < 0$.

Case 2. $S_{i,m} = 0$. We now consider $U_{i,m} \geq 1$. Let t_1 denote the first of these unsuccessful interactions within the time window Δt. For the second interaction request within the time window, i must have calculated the trust value for m as

$$\begin{aligned} T_{i,m} &= 100\left(\frac{S_{i,m}}{S_{i,m} + U_{i,m}}\right)\left(1 - \frac{1}{S_{i,m}+1}\right) \\ &= 100\left(\frac{0}{0+1}\right)\left(1 - \frac{1}{0+1}\right) = 0. \end{aligned}$$

However, this is a contradiction, since the lower bound for the uncertain zone is always higher than 0. This proves the claim.

Case 3: $S_{i,m} = 0$, $U_{i,m} = 0$. The same as Case 3 of Claim 1. □

The above two claims are proved under the constraints that the trust value lies between 0 and 100. For a variable upper limit of trust value, the claims still hold. Let T_u be the variable denoting the upper limit of trust value. Notice that the formula for time-based past interaction will change accordingly with the numeric value 100 replaced by T_u in (1). Let us also give generic limits for the initial value of the function f as f_u, which in the above was fixed at 25, and for the initial value of uncertain zone as R_u, which was previously fixed at 50. Assign a value of g_u to the initial value of g, which is now fixed at 17. In both Claims 1 and 2, Cases 2 and 3 obviously still hold. For Case 1, it is not hard to see that the claims hold with certain restrictions on T_u, f_u, R_u, and g_u. Let us first look at Case 1 of Claim 1: For $R_i \neq \varphi$, there are no constraints as T_u would cancel on both sides when replaced by the quantity 100 on both sides. For $R_i = \varphi$, we obtain

$$\begin{aligned} T_u - f_u &< T_u\left(\frac{1}{a+1} - \frac{1}{(a+1)(S_{i,m}+1)}\right) \\ \Rightarrow 1 - \frac{f_u}{T_u} &< \left(\frac{1}{a+1} - \frac{1}{(a+1)(S_{i,m}+1)}\right). \end{aligned}$$

Carrying with the same argument as in the claim, we get that for the contradiction $\frac{1}{S_{i,m}+1} < 0$ to hold we should have that: $\frac{f_u}{T_u} < \frac{1}{2}$, i.e., $f_u < \frac{T_u}{2}$. In other words, f_u should be fixed at less than half the value of T_u.

Moving on to Case 1 of Claim 2, first suppose that $M_i \neq \varphi$. We have that

$$R_u - T_u\left(\frac{1}{2a+1} - \frac{1}{(2a+1)(S_{i,m}+1)}\right) < \frac{R_u}{3}.$$

Now, for the contradiction $\frac{1}{S_{i,m}+1} < 0$ to hold with $a \geq 1$, after some algebraic manipulation we reach that: $R_u \geq \frac{T_u}{2}$. In other words, R_u should be at least half the value of T_u. For $M_i = \varphi$, we have that

$$R_u - g_u < T_u\left(\frac{1}{2a+1} - \frac{1}{(2a+1)(S_{i,m}+1)}\right).$$

Once again, since $a \geq 1$, we get after solving the inequalities that $\frac{1}{S_{i,m}+1} < 0$ will hold if the following condition is met: $g_u \leq R_u - \frac{T_u}{3}$. In other words, the upper limit of the untrusted zone should always be greater or equal to one third the value of T_u.

By dishonest behavior, we mean a node providing false information about another node. Notice that this information might be a higher trust value or a lower trust value than the actual trust value. We assume that all good nodes for a particular node will always remain honest, whereas bad nodes for a node might show dishonest behavior. A trust calculation method is said to be resilient against dishonest behavior if by simulating the bad and dishonest nodes in the algorithm by bad but honest nodes we get the same trust value.

Definition 5.7. *A set of bad nodes B_i for a node i is said to have successfully cheated i, if for a node j, the trust calculation algorithm "A" for j*

$$\mathrm{A}\Big(\Big\{T'_{x,j}|x \in \mathrm{B}_i\Big\}, \Big\{T'_{y,j}|y \in \mathrm{B}'_i\Big\}\Big) \\ \neq \mathrm{A}\big(\{T_{x,j}|x \in \mathrm{C}_i\}, \{T_{y,j}|y \in \mathrm{B}'_i\}\big),$$

where C_i is a set in which every bad node in B_i is replaced by an honest but bad node.

Claim 3. *Our GTMS is resilient against cheating at the node level.*

Proof. The proof is straightforward. The only point in our protocol where we need the trust values from the other nodes while calculating the trust value of a node is during peer recommendation. However, since we do not ask the recommendation from the bad nodes or the really bad nodes, therefore

$$\mathrm{A}\Big(\Big\{T'_{i,y}|y \in \mathrm{B}'_i\Big\}\Big) = \mathrm{A}\Big(\Big\{T'_{i,y}|y \in \mathrm{B}'_i\Big\}\Big),$$

as we assumed that the good nodes would always behave honestly. □

In the aforementioned text, we have attributed dishonest behavior (sending false recommendation values) to bad or really bad nodes for a particular node, say i. There might be nodes that are good nodes for i yet at the same time bad or really bad nodes for a node j. Whenever i wishes to find recommendations for j, this set of nodes might send false recommendations to i. Going further, we can even associate dishonest behavior to good nodes as well. If the number of such dishonest nodes is far less as compared to the honest ones, the effect of these false recommendations on the overall trust value as calculated by (5) would be minimum. However, a collaboration of a greater number of nodes will affect the trust value to a greater degree. This is true since (5) has the form of a weighted average measure. Thus, (5) has a slight built-in capability of diminishing the effect of abnormal recommendations. As we will see in the next sections, similar is true for trust calculation at the BS level.

There is another interesting way in which a collaboration of nodes might work together in achieving a malicious goal. Suppose we have nodes i, j, and k. Node j is within i's radio range, while node k is not. k, however, is in the radio range of j. i sends a data packet to j, which in turn sends the data packet to k. If k drops the packet, j should count that as an unsuccessful interaction. However, if j and k are collaborating, whereby j does not count it as an unsuccessful interaction, then there is no way that i would be able to detect it. Thus, i might continue to send packets to j, which in turn would send them to k, only to be dropped by it. This, however, can be resolved if i sends its packets uniformly at random to all its trusted neighboring nodes turn by turn. This way, i will not send every packet to the two collaborating nodes and much of its packets will be forwarded successfully provided there is not a high percentage of collaborating nodes among its neighbors. This will prohibit the above-mentioned scenario from reoccurring every time.

5.1.2 Resilience Analysis at Cluster Head Level

At the CH, the trust value is calculated by getting the trust states of all nodes. At this stage of the protocol, we check the behavior of a collaboration of really bad nodes. We assume that in a group with $n+1$ nodes including the cluster head, the number of really bad nodes are less than or equal to $\lfloor n/2 \rfloor$. These really bad nodes are really bad for all other nodes in the group.

Definition 5.8. *A set of really bad nodes (mal) are said to be collaborating with each other if they provide false trust states of a particular node to the cluster head.*

Definition 5.9. *A collaboration of really bad nodes is successful against a node $j \notin mal$, if the following conditions hold:*

1. $\forall i \notin mal$, $s_{i,j} = trusted$,
2. $Z_j < -1$.

Definition 5.10. *A collaboration of really bad nodes is successful internally for a node $m \in mal$, if the following conditions hold:*

1. $\forall i \notin mal$, $s_{i,m} = untrusted$,
2. $Z_m > 1$.

Claim 4. *A set of really bad nodes cannot collaborate successfully against a node $j \notin mal$ and internally for a node $m \in mal$.*

Proof. We have

$$Z_j = \frac{\sqrt{3}\Big(X(s_{ch,j}) + \sum_{i=1,i\neq j}^{n} X(s_{i,j}) - n\Big)}{\sqrt{n}}.$$

Now, $\sum_{i\notin mal} X_{i,j} \geq 2\lfloor n/2 \rfloor \geq n$. Therefore,

$$Z_j \geq \frac{\sqrt{3}(n-n)}{\sqrt{n}} \geq 0.$$

This shows that the cluster head will not label this node as an untrusted node. For part 2, notice that $\sum_{i\in mal,i\neq m} X_{i,m} \leq 2(\lfloor n/2 \rfloor - 1) \leq n-2$. Since $\forall i \notin mal, s_{i,m} = untrusted$, therefore

$$Z_m \leq \frac{\sqrt{3}(n-2-n)}{\sqrt{n}} \leq \frac{-2\sqrt{3}}{\sqrt{n}} < 0.$$

This implies that bad nodes would never make it to the trusted zone at the cluster head. □

Definition 5.11. *A group is said to be "malicious" if during its course of interactions with the other group the majority of interactions are unsuccessful.*

We will denote a malicious group by G_m. Let G denote the set of nodes in a generic group inside the sensor network.

Definition 5.12. *A malicious group G_m is said to have successfully deceived a group G_j, if for all groups $G_i \in G - G_m$, $s_{G_i,G_m} = trusted$ and there exists at least one $G_j \in G - G_m$, such that: $U_{G_i,G_m} \geq S_{G_i,G_m}$ and at least one of U_{G_i,G_m} and S_{G_i,G_m} is nonzero.*

Definition 5.13. *A Trust Management Scheme is said to be resilient against deception at group level if no group can successfully deceive another group.*

Claim 5. *Our GTMS is resilient against deception at group level.*

Proof. Similar to Claim 1. □

Definition 5.14. *A malicious group G_m is said to have partially deceived a group G_j, if for all groups $G_i \in G - G_m$, $s_{G_i,G_m} =$ uncertain and there exists at least one $G_j \in G - G_m$, such that: $U_{G_i,G_m} \geq 2S_{G_i,G_m}$ and at least one of U_{G_i,G_m} and S_{G_i,G_m} is nonzero.*

Definition 5.15. *A Trust Management Scheme is said to be resilient against partial deception at group level if no group can partially deceive another group.*

Claim 6. *Our GTMS is resilient against partial deception at group level.*

Proof. Similar to Claim 2. □

5.1.3 Resilience Analysis at Base Station Level

At the BS, the trust values of various groups are calculated. There can be three possible ways in which a particular group could cheat or try to get an unfair advantage. First, it might try to increase its own trust value even though it has not behaved well in the past. This cannot be done, as the BS asks other groups for their recommendations and its own past interaction records. Hence, the group whose trust value is being calculated has no say in this computation. The second scenario deals with one or more group nodes collaborating to harm the trust calculation of a particular group by submitting low but false recommendations for that group. Finally, these collaborating nodes might try to enhance each other's trust values at BS by giving high but false recommendations to the BS. We assume that the only group that will show dishonest behavior is this set of really bad groups.

Definition 5.16. *A set of bad groups B_i for the BS is said to have successfully cheated, if for a group j, the trust calculation algorithm "A" for j has the following property:*

$$\begin{aligned}&\mathrm{A}\Big(\Big\{T'_{x,j}|x \in \mathrm{B}_i\Big\}, \Big\{T'_{y,j}|y \in \mathrm{B}'_i\Big\}\Big)\\&\quad \neq \mathrm{A}\big(\{T_{x,j}|x \in \mathrm{C}_i\}, \{T_{y,j}|y \in \mathrm{B}'_i\}\big),\end{aligned}$$

where C_i is the set obtained by replacing every bad and dishonest group in B_i with a bad but honest group.

Claim 7. *Our GTMS is resilient against cheating at the BS.*

Proof. The proof is straightforward. The only place in our protocol where we need the trust values from the other nodes while calculating the trust value of a node is during peer recommendation. However, since the BS does not ask the recommendation from the bad groups, therefore

$$\mathrm{A}\Big(\Big\{T'_{i,y}|y \in B'_i\Big\}\Big) = \mathrm{A}\Big(\Big\{T'_{i,y}|y \in B'_i\Big\}\Big).$$

□

5.2 Communication Overhead

We assume a worst case scenario, in which every member node wants to communicate with every other node in the group and every group wants to communicate with the rest of the groups in the network. Let us assume that the network consists of $|G|$ groups and the average size of groups is σ.

TABLE 1
Communication Overhead in Worst Case

	Communication overhead
GTMS	$2|G|\ [\sigma(\sigma-1)(\sigma-2) + (|G|-1)]$
RFSN	$2|G|\ [\sigma(\sigma-1)(\sigma-2) + (|G|-1)(|G|-2)]$
PLUS	$2|G|\ \left[\sigma(\sigma-1)^2 + (|G|-1)^2\right]$
ATRM	$4|G|\ [\sigma(\sigma-1) + (|G|-1)]$

In the intragroup communication case, when node i wants to interact with node j, node i will send maximum $\sigma - 2$ peer recommendation requests. In response, node i will receive $\sigma - 2$ responses. If node i wants to interact with all the nodes in the group, the maximum communication overhead will be $2(\sigma-1)(\sigma-2)$. If all nodes want to communicate with each other, the maximum intragroup communication overhead (C_{intra}) of the GTMS is $2\sigma(\sigma-1)(\sigma-2)$.

In the intergroup communication case, when group i wants to interact with group j, it will send one peer recommendation request to the BS, at the maximum. So, the communication overhead is two packets. If group i wants to communicate with all the groups, then the maximum communication overhead will be $2|G| - 1$ packets. If all the groups want to communicate with each other, the maximum intergroup communication overhead (C_{inter}) of the GTMS is $2|G|(|G| - 1)$. Therefore, the maximum communication overhead (C) introduced by the GTMS in the network is

$$\begin{aligned}C &= |G| \times C_{intra} + C_{inter}\\C &= |G|[2\sigma(\sigma-1)(\sigma-2)] + 2|G|(|G|-1)\\C &= 2|G|[\sigma(\sigma-1)(\sigma-2) + (|G|-1)].\end{aligned} \tag{11}$$

In general, communication overhead introduced by the GTMS in the whole network is

$$C = 2|G|[\sigma(\sigma-1)\rho + (|G|-1)], \tag{12}$$

where ρ represents the average number of recommender nodes in the group. Communication overhead of other schemes is shown in Table 1. More details about the RFSN scheme, ATRM scheme, and PLUS are given in Appendix A.1.

5.2.1 Comparison

Fig. 4 shows the communication overhead of various trust management schemes for a large-scale WSN (10,000 nodes) having equal size of clusters. It shows that as the number of cluster increases in the network the GTMS introduces less communication overhead as compared to the other schemes. Also, it indicates that GTMS is suitable for large-scale WSNs having small size of clusters. The important thing that we need to note here about the ATRM scheme is that it shows the result of just one transaction of each node.

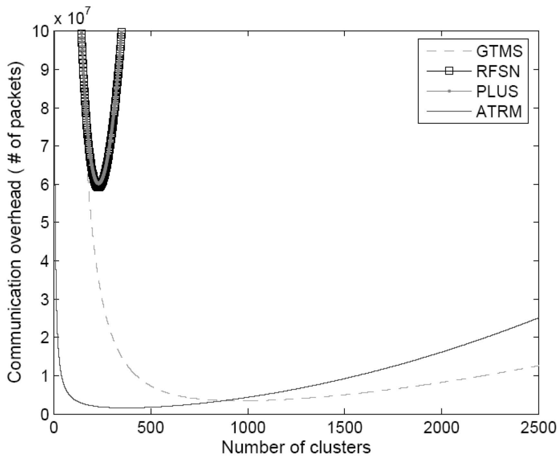


Fig. 4. Communication overhead: Number of $\mathrm{nodes} = 10,000$.

For example, when node i wants to communicate with node j they first exchange four packets. Once the transaction is completed and node i wants to initiate another transaction with j, then the trust will be computed again. So, the communication overhead of the ATRM scheme will increase with the factor of four with every transaction. Whereas for the case of the GTMS, after completion of the first transaction, when node i wants to start another transaction with j, no extra communication overhead will occur because node i will calculate the trust based on the history of past transaction(s).

5.3 Memory Consumption Analysis

One of the critical constraints of SNs is less availability of memory. For example, MICA2 SN has 128-Kbyte program flash memory, 512-Kbyte measurement flash, and 4-Kbyte EEPROM [39]. Our GTMS does conform to this low-memory demand as discussed below.

5.3.1 Memory Requirement of GTMS at Node Level

Each node maintains a small trust database as shown in Table 2. The size of each record is $4 + 4\Delta t$ bytes, where Δt represents the size of the time window. Therefore, memory requirement for GTMS at each SN is $(n - 1)(4 + 4\Delta t)$ bytes, where n represents the number of nodes in a group. The size of the trust table depends upon the size of the cluster and the length of time window.

5.3.2 Memory Requirement of GTMS at Cluster Head Level

Each CH maintains two tables; one is similar to an individual SN's trust table, and in the second, CH maintains the trust values of other groups as shown in Table 3. The size of each record is $4 + 4\Delta t$ bytes. Therefore, the total size of Table 3 is $(|G| - 1)(4 + 4\Delta t)$ bytes, where $|G|$ represents the number of groups in the network. The total memory space required at the CH for both tables is $(|G| + \sigma - 2)(4 + 4\Delta t)$ bytes. Here, σ represents the average number of nodes in the group.

TABLE 2
Trust Database at SN

Node ID	Past interactions based on time window						Peer recomm.	Trust value
	$S_{x,y}$			$U_{x,y}$				
	t_1	$\cdots$	t_n	t_1	$\cdots$	t_n		
2 bytes	2 bytes	$\dots$	2 bytes	2 bytes	$\dots$	2 bytes	1 byte	1 byte

TABLE 3
Group Trust Database at Cluster Head

Node ID	Past interactions with other groups based on time window						Peer recomm. from BS	Trust value
	$S_{x,y}$			$U_{x,y}$				
	t_1	$\cdots$	t_n	t_1	$\cdots$	t_n		
2 bytes	2 bytes	$\dots$	2 bytes	2 bytes	$\dots$	2 bytes	1 byte	1 byte

5.3.3 Comparison

In the simulation, we assumed that all clusters are of equal size. We set the time window Δt equal to 5. So, the size of trust record is 24 bytes. We have compared our scheme with the RFSN scheme [24], ATRM scheme [25], and PLUS [26] for the same clustering topology. Memory requirement of these schemes is given in Table 4, in which n represents the number of nodes in the group, N represents the total number of nodes in the network, and k represents the number of context. Details about how the memory requirements of the RFSN scheme, ATRM scheme, and PLUS are calculated are given in Appendix A.2.

Results in Fig. 5 are for 100 SNs. This graph shows that GTMS at SNs and CHs consumes less memory as compared to the ATRM scheme, PLUS, and RFSN scheme. Memory consumption of GTMS at the CH depends upon the number of clusters in the network. As the number of clusters increases, the memory consumption requirement also increases linearly at the CH. For example, if the network consists of 100 clusters with an average size of 20 nodes, then at the CH, GTMS consumes 2,832 bytes of memory. This shows that GTMS can be used for large-scale sensor networks.

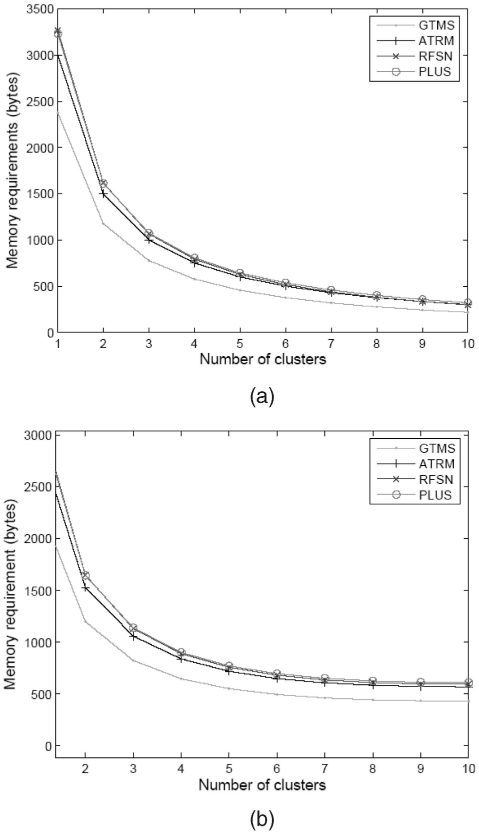
6 Simulation-Based Analysis and Evaluation

6.1 Simulation Environment

We have performed simulation using Sensor Network Simulator and Emulator (SENSE) [40]. We have deployed three different sized sensor networks consisting of 144, 225, and 324 SNs. More details about these networks are available in Table 5. Nodes are static and are organized in a grid fashion. The first, second, and third networks are comprised of 16, 25, and 36 clusters, respectively. These

TABLE 4
Memory Requirement of Trust Management Schemes

	Sensor node	cluster head
GTMS	$(n-1)(4+4\Delta t)$	$(\lvert G\rvert+\sigma-2)(4+4\Delta t)$
RFSN	$33(n-1)$	$33(G+\sigma-2)$
PLUS	$32.375(n-1)+28$	$32.375(G+\sigma-2)+28$
ATRM	$30n+8(k-1)$	$30(G+\sigma)+2(4k-19)$



(a)

(b)

Fig. 5. Memory requirement: $N = 100$ and $\Delta t = 5$ units. (a) At SN. (b) At cluster head.

numbers are chosen to make all clusters in equal size of nine nodes. Each network comprises of one BS that is located at the middle of the corresponding terrain. In all three networks, we used free space wireless channel, IEEE 802.11 MAC protocol, and a simplified version of DSR routing protocol (without route repairing). At the application layer, we have developed our own generic and simple Trust Exchange Protocol (TExP) that consists of six fields:

1. SourceID: contains the identity of the source node.
2. DestID: contains the identity of the destination node.
3. Protocol ID: represents the identity of the trust management protocol, e.g., GTMS, RFSN, and so forth.
4. Type: is used to identify the type of the packet such as request packet, response packet, acknowledgment packet, and so forth.
5. Payload: field is of variable size containing the data specific to the type and protocol, such as trust value, identity of evaluating node, and so forth.
6. SendT: contains the sending time of the packet.

TABLE 5
Sensor Network's Specifications

Network size	No. of clusters	Terrain
144 nodes	16	$600m \times 600m$
225 nodes	25	$800m \times 800m$
324 nodes	36	$1000m \times 1000m$

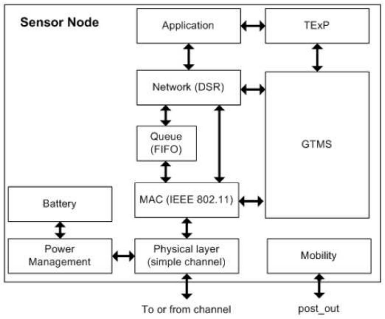


Fig. 6. SN architecture.

The objective of the TExP protocol is to exchange the trust values between communicating nodes in an efficient manner. SN architecture based on SENSE [40] is shown in Fig. 6, which shows the interactions between GTMS, TExP, and other components. The rest of the specifications of an SN is defined in Table 6.

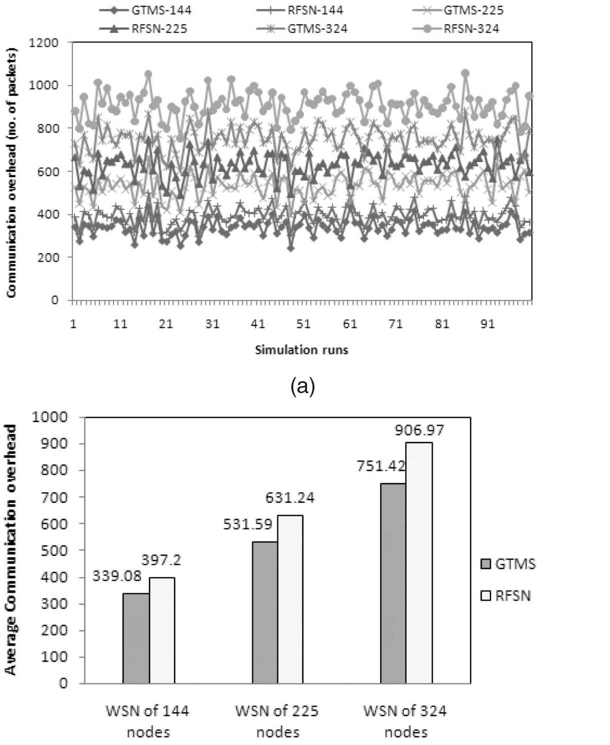
6.2 Comparison

For the purpose of comparison, we have implemented a peer recommendation scenario. During simulation, in each cluster, random number of source nodes are selected, which perform peer recommendation with the other nodes. Also, each cluster head will perform peer recommendation with neighboring cluster heads only. In the simulation, we have only compared our proposed GTMS with the RFSN scheme because both are independent of any specific routing scheme and platform. We did not implement the ATRM scheme because it requires some specific agent-based platform. Also, we did not implement PLUS because it works on the top of its own defined routing protocol.

Communication overhead for the three different networks is shown in Fig. 7, which confirms our conclusions from the theoretical analysis. Fig. 7a shows that the GTMS introduces less communication overhead as compared to the RFSN scheme, and this pattern (overhead difference) approximately remains the same for all 100 simulation runs. Therefore, we conclude that the 100 simulation runs can give us reliable results. Fig. 7b shows that, as the network size increases, the communication overhead difference between the GTMS and RFSN scheme also increases. It shows that the GTMS would introduce 14.6 percent, 15.7 percent, and 17.1 percent less communication overhead

TABLE 6
SN's Specifications

Initial battery of each sensor node	$1 \times 10^6 J$
Power consumption for transmission	$1.6W$
Power consumption for reception	$1.2W$
Power consumption in idle state	$1.15W$
Transmission power of the antenna	0.0280
Transmission and Reception gain	1.0
Carrier sense threshold	$3.652e^{-10}W$
Receive power threshold	$1.559e^{-11}W$



(a)

(b)

Fig. 7. Average communication overhead analysis (100 simulations). (a) Communication overhead. (b) Average communication overhead.

as compared to the RFSN scheme for the network of 144, 225, and 324 nodes, respectively.

Communication overhead also affects the energy consumption of the SNs. That effect is visible in Fig. 8, which shows that GTMS also consume less energy as compared to the RFSN scheme.

7 Conclusion and Future Directions

With the emergence of widespread use of WSNs, the need of a proper trust management scheme is strongly felt. In this work, we have proposed a robust lightweight GTMS for clustered WSNs. GTMS uses a hybrid trust management approach, which reduces the cost of trust evaluation. We showed that our scheme is memory efficient and consumes less communication overhead. We also proved that the GTMS is intrusion tolerant and provides protection against malicious, selfish, and faulty nodes.

In many application scenarios [41], [42], SN identities should remain hidden for achieving identity anonymity. So, the challenging problem is how to establish and maintain trust between communicating nodes in an identity anonymous environment. This motivates future work.

Appendix A

A.1 Communication Overhead

A.1.1 RFSN

When node i wants to interact with node j, it will send $n-2$ peer recommendation requests at the maximum. In response, node i will receive $n-2$ responses. If node i want to interact with all the nodes in the group, then the maximum communication overhead will be $2(n-1)(n-2)$. If all the nodes want to communicate with each other, the maximum intragroup communication overhead (C_{intra}) will be: $2n(n-1)(n-2)$. When the CH of group i wants to interact with the CH of group j, it will send $|G|-2$ peer recommendation requests at the most. So, the communication overhead will be: $2(|G|-2)$. If group i wants to communicate with all the groups, then the maximum communication overhead will be: $2(|G|-1)(|G|-2)$. If all the groups want to communicate with each other, then the maximum intergroup communication overhead (C_{inter}) will be: $2|G|(|G|-1)(|G|-2)$. Therefore, in the worst case, the maximum communication overhead (C) introduced by the RFSN scheme in the whole network is

$$\begin{aligned}
C &= |G| \times C_{intra} + C_{inter}\\
C &= |G|[2\sigma(\sigma-1)(\sigma-2)] + 2|G|(|G|-1)(|G|-2)\\
C &= 2|G|[\sigma(\sigma-1)(\sigma-2) + (|G|-1)(|G|-2)],
\end{aligned}$$

where σ represents the average number of nodes in the group, and $|G|$ represents the total number of groups in the network.

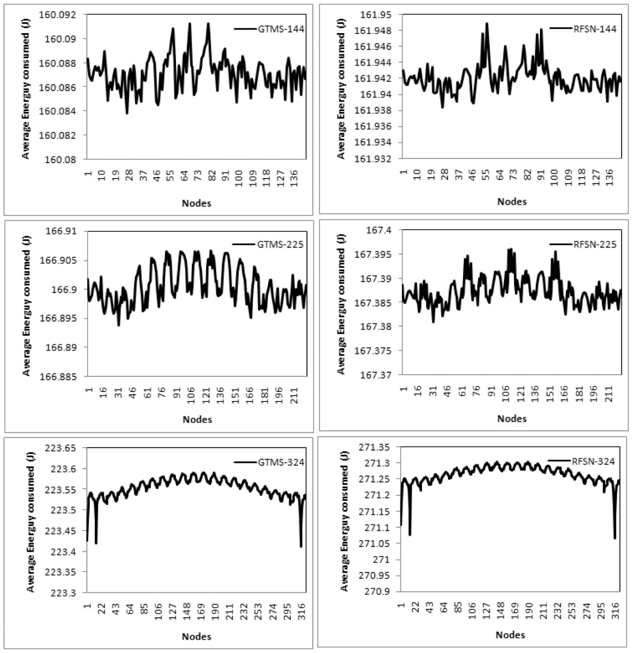


Fig. 8. Average energy consumption at each node (100 simulations).

A.1.2 PLUS

If node i wants to interact with another node j, then it will broadcast a request packet. In response, i will get $n-2$ responses. So, the communication overhead will be: $1+(n-2)$. If node i wants to communicate with all the nodes in the group, the communication overhead will be: $(n-1)+(n-1)(n-2)$. If all the nodes want to communicate with each other, then the total intragroup communication overhead (C_{intra}) will be: $n(n-1)+n(n-1)(n-2)=n(n-1)^2$.

Each node in the group can also exchange antiactive protocol, whose communication cost is the same as getting recommendation from other nodes. So, in the worst case, the total intragroup communication overhead (C_{intra}) will be: $2n(n-1)^2$. If group i wants to interact with another group j, then group i will broadcast a request packet. In response, it will get no more than $|G|-2$ responses. So, the communication overhead will be: $1+(|G|-2)$. If group i wants to communicate with all the groups, then maximum communication overhead will be: $(|G|-1)+(|G|-1)(|G|-2)$. If all the groups want to communicate with each other, the total intergroup communication overhead (C_{inter}) will be

$$C_{inter} = |G|(|G|-1) + |G|(|G|-1)(|G|-2)$$
$$C_{inter} = |G|(|G|-1)[1+(|G|-2)] = |G|(|G|-1)^2.$$

If we add the communication overhead of antiactive protocol, then the maximum communication overhead for intergroup (C_{inter}) will be: $2|G|(|G|-1)^2$. Therefore, in the worst case, the maximum communication overhead (C) introduced by PLUS in the whole network is

$$C = |G|C_{intra} + C_{inter} = |G|2\sigma(\sigma-1)^2 + 2|G|(|G|-1)^2$$
$$C = 2|G|\left[\sigma(\sigma-1)^2 + (|G|-1)^2\right].$$

A.1.3 ATRM

Each node needs to exchange four packets in order to compute the trust. If a node i wants to communicate with all the nodes in the group, then the communication overhead will be: $4(n-1)$. If all the nodes want to communicate with each other, then the total communication overhead (C_{intra}) will be: $4n(n-1)$. Similarly, if all groups want to communicate with each other, the intergroup communication (C_{inter}) will be: $4|G|(|G|-1)$. Therefore, in the worst case, the maximum communication overhead (C) introduced by the ATRM scheme in the whole network is

$$C = |G|C_{intra} + C_{inter} = |G|4\sigma(\sigma-1) + 4|G|(|G|-1)$$
$$C = 4|G|[\sigma(\sigma-1) + (|G|-1)].$$

A.2 Memory Consumption

A.2.1 RFSN

Each SN also needs to store two tables: Reputation Module Matrix (RMM) and RFSN monitor. RMM consist of eight parameters: Context (4 bytes), Aging period (4 bytes), Aging weight (4 bytes), Integration weight (4 bytes), Size (1 byte), Alpha (4 bytes), Beta (4 bytes), and Node ID (2 bytes). So, the size of one record in RMM is 27 bytes. RFSN monitor maintains two parameters: Node ID (2 bytes) and Data readings (4 bytes). So, the size of one record of RFSN monitor is 6 bytes. Thus, the total memory required by the RFSN scheme at SN is

$$M_{SN} = \text{size(RMM)} + \text{size(Monitor)}$$
$$M_{SN} = 27(n-1) + 6(n-1) = 33(n-1).$$

Here, n represents the number of nodes in the neighborhood. Let us assume that every CH also maintains the trust value of other CHs in the same manner as nodes maintain trust value of other member nodes. Then, memory requirement at the CH in the RFSN scheme is

$$M_{CH} = 33(|G|-1) + 33(\sigma-1) = 33(|G|+\sigma-2).$$

A.2.2 ATRM

Each SN stores two tables: Trust evaluation table (Tab_{eval}) and t_Instrument table (Tab_{instr}). The Tab_{eval} table consists of four parameters: Node ID (2 bytes), Trust Context (4 bytes), Evaluation (4 bytes), and Time stamp (4 bytes). So, the size of each record is 14 bytes. The Tab_{instr} table consists of five parameters: Node ID (2 bytes), Trust context (4 bytes), INSTR (4 bytes), Time stamp (4 bytes), and ACK (2 bytes). So, the size of each record for Tab_{instr} table is 16 bytes. Each SN also stores the r_certificate (r_{cert}) in a memory. The size of certificate varies with respect to the number of available contexts. The r_certificate is defined as: $RC = E_{AK}(R, H(R))$, where $R = ID_i$, T, $((r_1, C_1), (r_2, C_2), \ldots, (r_k, C_k))$. Here, ID represents the identity of the node (2 bytes), T represents the time stamp (4 bytes), and r_1 (4 bytes) represents the reputation of node i under context c_1 (4 bytes). So, the size of R is $6+8k$. If we assume the MD5 hash function (16 bytes), then the total size of r_{cert} is $22+8k$. Thus, the total memory required at the SN is

$$M_{SN} = \text{size}(\text{Tab}_{\text{eval}}) + \text{size}(\text{Tab}_{\text{instr}}) + \text{size}(\text{r\_cert})$$
$$M_{SN} = 14(n-1) + 16(n-1) + (22+8k)$$
$$M_{SN} = 30(n-1) + 22 + 8k = 30n + 8(k-1).$$

Let us assume that every CH also maintains the trust value of other CHs in the same manner as nodes maintain the trust value of other member nodes. CH maintains a single r_cert that is used for inter- and intracommunications. That is why the size of certificate will be added once. Thus, in this case, memory requirement at the CH is

$$M_{CH} = 30(|G|-1) + 30(\sigma-1) + 22 + 8k$$
$$M_{CH} = 30(|G|+\sigma-2) + 22 + 8k$$
$$= 30(|G|+\sigma) + 2(4k-19).$$

A.2.3 PLUS

In the case of PLUS, each SN needs to store two tables and seven constant context parameters. The first table consists of node ID (2 bytes), personal reference parameters (T_{or} (1 bit), T_{ai} (1 bit), T_{ce} (1 bit), T_{po} (4 bytes), T_{re} (4 bytes), T_{co} (4 bytes)), peer recommendation value T_i (4 bytes), and final calculated trust value (4 bytes). So, the size of one record of the first table is 22.375 bytes. In the second table, a node needs to store information about node ID (2 bytes), number of requests sent (2 bytes), number of reply received (2 bytes), number of packets actually forwarded (2 bytes), and number of packets supposed to be forwarded (2 bytes). So, the size of one record for the second table is 10 bytes. Each of the context parameters $(W_{cp}, W_{po}, W_{re}, W_{oo}, W_{av}, W_{pr}, W_r)$ is represented by 4 bytes. So, the total size required to store context parameters is 28 bytes. Thus, the total memory required at the SN is

$$M_{SN} = \text{size(table1)} + \text{size(table2)} + \text{contextParameters}$$
$$M_{SN} = 22.375(n-1) + 10(n-1) + 28 = 32.375(n-1) + 28.$$

Let us assume that every CH also maintains the trust value of other CHs in the same manner as nodes maintain trust values of other member nodes. Then, in this case, memory requirement at the CH is

$$M_{CH} = 32.375(|G|-1) + 32.375(\sigma-1) + 28$$
$$M_{CH} = 32.375(|G|+\sigma-2) + 28.$$

Acknowledgments

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2009-(C1090-0902-0002)) and was supported by the IT R&D program of MKE/KEIT [10032105, Development of Realistic Multiverse Game Engine Technology]. This work also was supported by the Brain Korea 21 projects and Korea Science & Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. 2008-1342). The corresponding author is Sungyoung Lee.

References

[1] L.J. Hoffman, K. Lawson-Jenkins, and J. Blum, "Trust beyond Security: An Expanded Trust Model," *Comm. ACM,* vol. 49, no. 7, pp. 95-101, July 2006.

[2] E. Shi and A. Perrig, "Designing Secure Sensor Networks," *IEEE Wireless Comm.,* vol. 11, no. 6, pp. 38-43, 2004.

[3] H.S. Ng, M.L. Sim, and C.M. Tan, "Security Issues of Wireless Sensor Networks in Healthcare Applications," *BT Technology J.,* vol. 24, no. 2, pp. 138-144, Apr. 2006.

[4] A.A. Pirzada and C. McDonald, "Establishing Trust in Pure Ad-Hoc Networks," *Proc. 27th Australasian Computer Science Conf. (ACSC '04),* pp. 47-54, Jan. 2004.

[5] Y.L. Sun, W. Yu, Z. Han, and K.J.R. Liu, "Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, no. 2, pp. 305-317, Feb. 2006.

[6] R.A. Shaikh, H. Jameel, S. Lee, S. Rajput, and Y.J. Song, "Trust Management Problem in Distributed Wireless Sensor Networks," *Proc. 12th IEEE Int'l Conf. Embedded Real-Time Computing Systems and Applications (RTCSA '06),* pp. 411-414, Aug. 2006.

[7] M. Momani, S. Challa, and K. Aboura, "Modelling Trust in Wireless Sensor Networks from the Sensor Reliability Prospective," *Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecomm.,* T.S. et al., ed., pp. 317-321, Springer, 2007.

[8] J.P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Chapter 16: Wireless Sensor Network Security: A Survey," *Security in Distributed, Grid, and Pervasive Computing,* Y. Xiao, ed., pp. 367-410, CRC Press, 2006.

[9] Z. Liu, A.W. Joy, and R.A. Thompson, "A Dynamic Trust Model for Mobile Ad Hoc Networks," *Proc. 10th IEEE Int'l Workshop Future Trends of Distributed Computing Systems (FTDCS '04),* pp. 80-85, May 2004.

[10] S. Buchegger and J.-Y.L. Boudec, "Self-Policing Mobile Ad Hoc Networks by Reputation Systems," *IEEE Comm. Magazine,* vol. 43, no. 7, pp. 101-107, July 2005.

[11] T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," *IEEE Comm. Surveys and Tutorials,* vol. 3, no. 4, 2000.

[12] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Comm.,* vol. 1, no. 4, pp. 660-670, Oct. 2002.

[13] S. Lindsey, C. Raghavendra, and S. Raghavendra, "PEGASIS—Power-Efficient Gathering in Sensor Information Systems," *Proc. IEEE Aerospace Conf.,* vol. 3, pp. 1125-1130, 2002.

[14] A. Manjeshwar and D.P. Agrawal, "TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks," *Proc. 15th Int'l Parallel and Distributed Processing Symp. (IPDPS '01),* pp. 2009-2015, Apr. 2001.

[15] O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-Hoc Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 3, no. 4, pp. 366-379, Oct. 2004.

[16] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge," *IEEE Trans. Dependable and Secure Computing,* vol. 3, no. 1, pp. 62-77, Jan.-Mar. 2006.

[17] M. Shehab, E. Bertino, and A. Ghafoor, "Efficient Hierarchical Key Generation and Key Diffusion for Sensor Networks," *Proc. Second Ann. IEEE Conf. Sensor and Ad Hoc Comm. and Networks (SECON '05),* pp. 197-213, Sept. 2005.

[18] S. Bandyopadhyay and E.J. Coyle, "Minimizing Communication Costs in Hierarchically-Clustered Networks of Wireless Sensors," *Computer Networks,* vol. 44, no. 1, pp. 1-16, 2004.

[19] M. Gupta, P. Judge, and M. Ammar, "A Reputation System for Peer-to-Peer Networks," *Proc. 13th Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '03),* pp. 144-152, June 2003.

[20] D. Ingram, "An Evidence Based Architecture for Efficient, Attack-Resistant Computational Trust Dissemination in Peer-to-Peer Networks," *Proc. Third Int'l Conf. Trust Management,* pp. 273-288, May 2005.

[21] L. Xiong and L. Liu, "Peer Trust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 7, pp. 843-857, July 2004.

[22] G. Theodorakopoulos and J.S. Baras, "On Trust Models and Trust Evaluation Metrics for Ad Hoc Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, no. 2, pp. 318-328, Feb. 2006.

[23] S. Buchegger and J.-Y.L. Boudec, "A Robust Reputation System for Peer-to-Peer and Mobile Ad-Hoc Networks," *Proc. Second Workshop Economics of Peer-to-Peer Systems (P2PEcon '04),* June 2004.

[24] S. Ganeriwal and M.B. Srivastava, "Reputation-Based Framework for High Integrity Sensor Networks," *Proc. ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04),* pp. 66-67, Oct. 2004.

[25] A. Boukerche, X. Li, and K. EL-Khatib, "Trust-Based Security for Wireless Ad Hoc and Sensor Networks," *Computer Comm.,* vol. 30, pp. 2413-2427, Sept. 2007.

[26] Z. Yao, D. Kim, and Y. Doh, "PLUS: Parameterized and Localized Trust Management Scheme for Sensor Networks Security," *Proc. Third IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS '06),* pp. 437-446, Oct. 2006.

[27] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang, "Location Verification and Trust Management for Resilient Geographic Routing," *J. Parallel and Distributed Computing,* vol. 67, no. 2, pp. 215-228, 2007.

[28] H. Chen, H. Wu, X. Zhou, and C. Gao, "Reputation-Based Trust in Wireless Sensor Networks," *Proc. Int'l Conf. Multimedia and Ubiquitous Eng. (MUE '07),* pp. 603-607, Apr. 2007.

[29] H. Jameel, L.X. Hung, U. Kalim, A. Sajjad, S. Lee, and Y.-K. Lee, "A Trust Model for Ubiquitous Systems Based on Vectors of Trust Values," *Proc. Third IEEE Int'l Security in Storage Workshop (SISW '05),* pp. 674-679, Dec. 2005.

[30] R.A. Shaikh, S. Lee, M.A.U. Khan, and Y.J. Song, "LSec: Lightweight Security Protocol for Distributed Wireless Sensor Network," *Proc. 11th IFIP Int'l Conf. Personal Wireless Comm. (PWC '06),* pp. 367-377, Sept. 2006.

[31] A. Hac, *Wireless Sensor Network Designs.* John Wiley & Sons, 2003.

[32] R. Shah and J. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '02),* pp. 350-355, 2002.

[33] S. Muruganathan, D. Ma, R. Bhasin, and A. Fapojuwo, "A Centralized Energy-Efficient Routing Protocol for Wireless Sensor Networks," *IEEE Comm. Magazine,* vol. 43, no. 3, pp. 8-13, 2005.

[34] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks,* vol. 8, no. 5, pp. 521-534, 2002.

[35] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* pp. 62-72, 2003.

[36] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Symp. Security and Privacy,* pp. 197-213, May 2003.

[37] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," *Proc. Second Int'l Conf. Embedded Networked Sensor Systems (SenSys '04),* pp. 162-175, Nov. 2004.

[38] H. Tijms, *Understanding Probability: Chance Rules in Everyday Life.* Cambridge Univ. Press, 2004.

[39] Xbow, mica2 series, http://www.xbow.com, 2008.

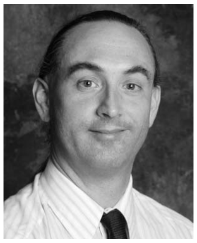
[40] B.K. Szymanski, *SENSE: Sensor Network Simulator and Emulator,* http://www.ita.cs.rpi.edu/sense/index.html, 2008.

[41] S. Olariu, Q. Xu, M. Eltoweissy, A. Wadaa, and A.Y. Zomaya, "Protecting the Communication Structure in Sensor Networks," *Int'l J. Distributed Sensor Networks,* vol. 1, pp. 187-203, 2005.

[42] S. Misra and G. Xue, "Efficient Anonymity Schemes for Clustered Wireless Sensor Networks," *Int'l J. Sensor Networks,* vol. 1, nos. 1/2, pp. 50-63, 2006.

Riaz Ahmed Shaikh received the BS degree in computer engineering from Sir Syed University of Engineering and Technology (SSUET), Karachi, Pakistan, in 2003 and the MS degree in information technology from the National University of Sciences and Technology (NUST), Rawalpindi, Pakistan, in 2005. He is currently a PhD candidate in the Department of Computer Engineering, Kyung Hee University, Suwon, Korea. His research interests include privacy and security and trust management. He is a professional member of the ACM. More information about him is available at http://member.acm.org/riaz289.

Hassan Jameel received the BE degree in computer software engineering from the National University of Sciences and Technology (NUST), Rawalpindi, Pakistan, in 2003 and the MS degree in computer engineering from Kyung Hee University, Suwon, Korea, in 2005. He is currently a PhD candidate in the Department of Computer Engineering, Kyung Hee University. His research interests include cryptography and trust management.



Brian J. d'Auriol received the BSc(CS) and PhD degrees from the University of New Brunswick in 1988 and 1995, respectively. He is currently with the Department of Computer Engineering, Kyung Hee University, Suwon, Korea. Previously, he had been a researcher at the Ohio Supercomputer Center, Columbus and an assistant professor at the University of Texas, El Paso, the University of Akron, Wright State University, and the University of Manitoba. He has organized and chaired the International Conference on Communications in Computing (CIC) from 2000 to 2008 and the 11th Annual International Symposium on High Performance Computing Systems (HPCS) in 1997. His research includes information and data visualization with specialization in software, bioinformatics, and healthcare visualizations; optical bus parallel computing models, and recently, ubiquitous sensor networks. He is a member of the ACM and the IEEE Computer Society.

Heejo Lee received the BS, MS, and PhD degrees in computer science and engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea. He is an associate professor in the Division of Computer and Communication Engineering, Korea University, Seoul. Before joining Korea University, he was a CTO at AhnLab from 2001 to 2003. From 2000 to 2001, he was a postdoctoral fellow in the Department of Computer Sciences and the Center for Education and Research in Information Assurance and Security (CERIAS), Purdue University. He has been serving as an editor of the *Journal of Communications and Networks* since 2007. He has been an advisory member of the Korea Information Security Agency and Korea Supreme Prosecutor's Office. With the support of the Korean government, he was working for constructing the National CERT in the Philippines (2006) and the consultation of Cyber Security in Uzbekistan (2007). More information is available at http://ccs.korea.ac.kr. He is a member of the IEEE.

Sungyoung Lee received the BS degree from Korea University, Seoul and the MS and PhD degrees in computer science from Illinois Institute of Technology (IIT), Chicago, in 1987 and 1991, respectively. He has been a professor in the Department of Computer Engineering, Kyung Hee University, Suwon, Korea, since 1993. He is a founding director of the Ubiquitous Computing Laboratory and has been a director of the Neo Medical Ubiquitous-Life Care Information Technology Research Center, Kyung Hee University since 2006. Before joining Kyung Hee University, he was an assistant professor in the Department of Computer Science, Governors State University, University Park, Illinois, from 1992 to 1993. His current research focuses on ubiquitous computing and applications, context-aware middleware, sensor operating systems, real-time systems, and embedded systems. He is a member of the ACM and the IEEE.

Young-Jae Song received the BE degree from Inha University, Incheon, Korea, in 1969, the MS degree in computer science from Keio University, Tokyo, in 1976, and the PhD degree from Myongji University, Seoul, in 1980. He has been a professor in the Department of Computer Engineering, Kyung Hee University, Suwon, Korea, since 1976. His current research focuses on software engineering, reverse engineering, component-based software development, and object-based modeling.