Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 952905, 14 pages
http://dx.doi.org/10.1155/2013/952905

Research Article

Two-Way Acknowledgment-Based Trust Framework for Wireless Sensor Networks

X. Anita,[1] J. Martin Leo Manickam,[2] and M. A. Bhagyaveni[1]

[1] *Department of ECE, Anna University, Chennai 600025, India*
[2] *Department of ECE, St. Joseph's College of Engineering, Chennai 600119, India*

Correspondence should be addressed to X. Anita; anitaextee@yahoo.co.in

Received 1 November 2012; Revised 3 April 2013; Accepted 4 April 2013

Academic Editor: Yanmin Zhu

Security is an inherent and challenging task in Wireless Sensor Networks (WSNs) characterized by multihop routing and stringent resource constraints such as limited processing capability, storage capacity, communication bandwidth, and energy. Most of the existing trust-based routing schemes require the support of promiscuous mode of operation and gather a large number of recommendations from the neighbors for trust derivation. These result in higher energy consumption, higher communication overhead, and larger memory requirements in a resource constrained sensor network. In this paper, we propose a new two-way acknowledgment-based trust (2-ACKT) framework with individual (2-ACKT-I) and group (2-ACKT-G) acknowledgment to minimize the communication overhead and memory requirement for trust establishment in WSNs. The 2-ACKT protocol calculates the direct trust using a link layer acknowledgment and a two-hop acknowledgment from a downstream neighbor. The simulation and theoretical results indicate that 2-ACKT scheme significantly outperforms the conventional multihop and trust-based routing schemes in terms of packet delivery ratio, network lifetime, communication overhead, and memory requirements.

1. Introduction

Wireless Sensor Networks (WSNs) consist of densely deployed tiny sensor nodes to monitor the real world environment by sensing, processing, and communicating about the sensor field [1]. In WSNs, the environment being monitored does not require any installed infrastructure for energy and communication thereby simplifying the system design and operation [2]. Sensor nodes (SNs) in WSNs suffer from resource constraints such as low computational capability, limited storage capacity, limited communication bandwidth, and the use of insecure communication channel [3]. WSNs are prone to varied types of attacks [3–8] such as black hole attack, sink hole attack, and sniffing attack, which require a security solution. The memory and power constrained sensors make the employment of cryptographic solutions such as centralized keying and pair wise key sharing techniques, less attractive security solutions for WSNs. Moreover, cryptographic solutions can successfully defend against outsider attack but can fail under insider malicious attacks [9]. This vulnerability along with the cooperative nature of sensor networks requires the need for assessing the trust relationships among the nodes in the network [10]. Trust is the level of confidence in an entity and classified as direct and indirect trust [11]. The direct trust is the trust derived based on the node A's experience with its neighbor B while the indirect trust is the trust derived by the node A based on the experience of its neighbors with node B. Several trust-based routing protocols proposed in the literature use direct or indirect trust or both to thwart malicious attacks. But for accurate derivation of direct trust, the participating sensors need to support the promiscuous mode of operation. The promiscuous mode of operation demands the sensor to be in the idle listening state till the next hop neighbor forwards the packet, and, hence, consumes more energy. Moreover, the promiscuous mode of operation does not always provide sufficient evidence on the behavior of a monitored node. A monitored node may not be able to relay the packet due to the low quality of the wireless link. Alternatively, the indirect trust is derived based on the recommendations gathered

from the neighbors. The large number of recommendations gathered from the neighbors increases the overhead and energy consumption in the network.

In order to minimize the overhead and energy consumption, we propose a two-way acknowledgment-based trust (2-ACKT) framework for WSN. The 2-ACKT model does not require the sensors to support the promiscuous mode of operation. It uses the link layer acknowledgments and also exploits the dense nature of the network to derive the trust on the neighbors. The main aim of the proposed 2-ACKT model is to meet the following design objectives:

(i) to minimize the number of recommendations gathered from the neighbors for trust evaluation thereby minimizing the control overhead,

(ii) to avoid the promiscuous mode of operation of sensors thereby allowing the sensors to be in sleep mode as directed by medium access control layer thereby reducing the energy consumption, and

(iii) to reduce the memory requirement by representing the trust with lower number of bits.

This paper is organized as follows: Section 2 describes the related work. Section 3 presents the proposed 2-ACKT protocol. Section 4 discusses performance analysis. Section 5 offers conclusions and future scope of the work.

2. Related Work

Many trust-based routing schemes proposed for WSNs use either direct observation or recommendations or a combination of both to derive trust on their neighbors. Trust-based routing protocol TRANS [12] was proposed to thwart black hole attack [3]. It used a light-weight broadcast management scheme μTESLA [13] to authenticate all requests. The drawback of this protocol was that it required Message Authentication Code (MAC) along with shared encryption key to ensure integrity and confidentiality of request in routing. k-parent Flooding Trust Model (k-FTM) [14, 15] was robust against black hole attack on broadcast messages. It was a novel trust-aware framework designed specifically for performing secure broadcast communication in WSNs. The drawback of this approach was that it required a secure key-management protocol to establish pair-wise keys between each node and the base station. Ganeriwal and Srivastava proposed Reputation-based Framework for Sensor Network (RFSN) [16] scheme was a distributed reputation management approach. It employed a watchdog mechanism to build reputation and Beta reputation system to represent the reputation as well as to update the reputation continuously based on recommendations received from the neighbors. RFSN encountered bad mouthing attack [17] but at the cost of system efficiency as nodes could not share their bad experiences with others. Moreover, it also used past behavior of a node to predict its future behavior. This could be utilized by adversaries that planned attacks considering the weaknesses in different building blocks of the framework. The intelligent adversary could use a highly reputed node to abuse the system by potentially compromising it. Some work available in the literature [18, 19] used weight factor while integrating the direct and indirect trust value to avoid the bad mouthing attacks.

A distributed trust model Parameterized and Localized trUst management Scheme (PLUS) [20] was proposed to calculate trust by direct and indirect observations. It was a protocol specific trust mechanism and worked on the top of Parameterized Localized trUst management based Secure Routing (PLUS-R). In this scheme, the important control packets generated by the base station (BS) were appended with a hashed sequence number (HSN) to check the identity of the BS. Inclusion of this HSN increased the overhead of communication and computation. Trust values were updated for node i by the judge node on receiving a packet from node i. On reception, judge node would check the integrity of the packet. Integrity was checked by attaching MAC to thwart modification attack. If the integrity check failed, then the trust value of node i would decrease even though the modification was not performed intentionally thereby increasing the probability of false alarm. The judge node identified the black hole by setting its transceiver in promiscuous mode after routing some packets through the suspect and checked whether the suspect forwarded the packets. Trust value was represented in the range of 0 and 1. Using mobile agents, Agent-based Trust and Reputation Management (ATRM) [21] scheme maintained trust of each node in clustered WSNs. The trust was evaluated based on the service provided, that is, data forwarding behavior of the neighbor. The trust of the neighbor was updated by each node to its mobile agent. It assumed the existence of a trusted authority to generate and launch mobile agents and so it was vulnerable to a single point of failure. Also, it assumed that mobile agents were resilient against malicious attacks which were not realistic in many applications.

Agent-based Trust management model for WSNs (ATSN) [22] was a distributed agent-based trust mechanism that used promiscuous mode to overhear the behavior of its neighboring nodes and computed the trust. After computing the trust, the trust values were encrypted by the agent nodes and were broadcast to its neighbors. Li et al. proposed a novel scheme [23] that avoids bad recommendations by performing a deviation test by the monitoring nodes before accepting the indirect recommendation. The drawback of this scheme was that the monitoring nodes were assumed to be always trusted which was infeasible in real circumstances. Efficient Monitoring Procedure In a REputation system (EMPIRE) [24] was designed to reduce power consumption, memory usage, and communication overhead in WSNs. The main task of EMPIRE was to reduce the nodal monitoring activity and to maintain the system performance at a satisfactory level. It evaluated reputation based on direct and indirect observation. The reputation was computed by monitoring the packet forwarding behavior of a node and it thwarted black hole attacks. Reliable Adaptive serviCe-driven Efficient Routing (μRACER) [25] was proposed for sensor actuator network. It was based on three protocols namely, trust-aware routing protocol (TARP), context-aware routing protocol (CARP), and service-aware routing protocol (SARP). TARP was used to avoid routing through non-cooperative nodes

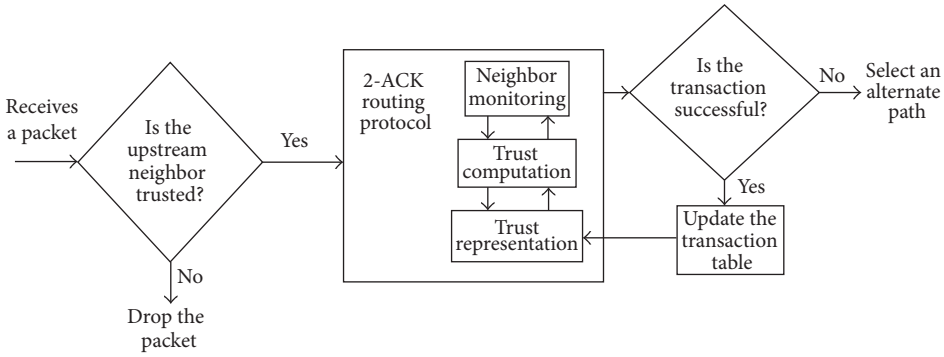

Figure 1: Block diagram of 2-ACKT routing protocol.

and to monitor the behavior of nodes. It considered both direct and indirect reputation. The scheme addressed the grey hole attack [3] by eliminating distrusted nodes from the routing path. To evaluate direct reputation, the scheme overheard the packet forwarding behavior of the neighbor.

Most of the trust management schemes did not address the various resource constraint requirements of WSNs. Therefore, Group-based Trust Management Scheme (GTMS) [26] overcame some of these constraints. Each node calculated trust based on direct or indirect observations. It adopted distributed trust management approach in intra group level and centralized trust management approach in inter group level. In intra group level, each node calculated trust by getting recommendations from all its group members, whereas in inter group level, each cluster head (CH) obtained recommendation about other CHs only from the BS. Trust value was represented in the range of 0 and 100, that is, as an unsigned integer that saved memory space and it addressed black hole attacks. The drawback of GTMS was the use of promiscuous mode of operation to monitor the neighbor. It demanded high energy and more memory space for CHs. Moreover, it also assumed a trusted BS which is immune to security threats.

Trust Management Architecture (TMA) [27] used super node-based trust management approach to overcome black hole attacks. It calculated trust based on direct and indirect observation. The direct observation was performed by monitoring the neighbor in promiscuous mode. The trust value representation and the assumptions made were similar to that of GTMS. Trust-based Cross Layer Model (TCLM) [28] scheme calculated trust by a cross-layer concept, that is, by using acknowledgments from data link layer and transport layer. The trust value was defined by Bayesian statistics and Beta distribution. The trust value was represented in the range of 0 and 1 as real numbers which required more memory. The aforementioned trusted routing schemes used promiscuous mode of operation to observe the packet forwarding behavior of a neighboring node. They have the following drawbacks:

(i) need for omni-directional transceivers;

(ii) compulsory support from sensors for full duplex transmission mode;

(iii) excessive energy consumption.

3. 2-ACKT Protocol

Our proposed 2-ACKT protocol calculated the direct trust based on the link layer acknowledgment (LLACK) and a two hop acknowledgment from the downstream neighbor.

The LLACK, which was derived from IEEE 802.15.4 MAC protocol, was a low rate wireless personal access network with low power consumption, low data rate and low cost requirement. The trust was computed by the SN based on the data forwarding behavior of a neighbor. The trust value was calculated from the number of successful and failed transactions. Here, transaction meant the cooperation of two nodes in forwarding the data packets. The sender would consider a transaction as successful if the sender confirmed that the packet was successfully received by the neighbor node and that node had forwarded the packet toward the destination honestly following the routing protocol. The 2-ACKT protocol consisted of four components such as neighbor monitoring, trust computation, trust representation and 2-ACK routing protocol as shown in Figure 1.

3.1. Assumptions. Trust is a relationship associated between two nodes for specific action, that is, data forwarding. Basically, each node trusts other nodes to perform data forwarding action. In this paper, the following terms are used to define the entities:

subject (S) is the SN that calculates the trust on its downstream neighbor in the discovered path to the BS,

target (T) is the downstream neighbor of the subject in the discovered path to the BS,

sponsor (R) is the downstream neighbor of the target in the discovered path to the BS,

third party (P) is a neighbor common to both the subject and sponsor.

The 2-ACKT protocol was designed with the following assumptions:

(i) all nodes behaved legitimately during route discovery stage,

(ii) each SN in the peer-to-peer network must have unique identity,

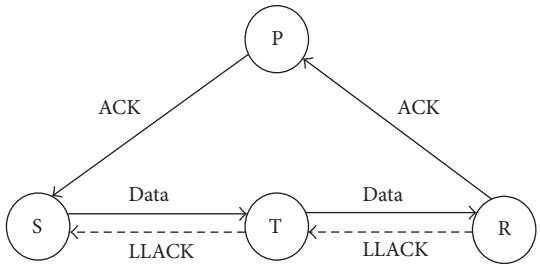

Figure 2: Neighbor monitoring.

(iii) all nodes were static and homogenous with regard to storage capacity, processing speed, and energy,

(iv) collaborative attackers were not present, and

(v) a secure communication channel which can be established with the help of any key management schemes [13, 29–31], provided protection for acknowledgments and data packets from traffic analysis or fabrication during transfer from one node to another.

3.2. Neighbor Monitoring. The neighbor monitoring component in a trust-based routing scheme must ensure that the neighbor has successfully received the packet and it has forwarded the packet honestly to its neighbor by following the underlying routing protocol. Consider the topology shown in the Figure 2. In 2-ACKT scheme, each node derives trust based on the packet forwarding behavior of its neighbor. Let us assume that the subject (S) unicasts a data packet to its neighbor target (T). Being a legitimate node, target will forward the packet to its next hop sponsor (R) by following the underlying routing protocol. Any packet forwarding misbehavior of the target reduces its trustworthiness in the network. For accurate derivation of trust, the subject must ensure the occurrence of the following two events:

(1) target has successfully received the packet sent by it, and

(2) target has forwarded the packet to its downstream neighbor R faithfully following the protocol.

In 2-ACKT scheme, the occurrence of *event (1)* was ensured by using the link layer acknowledgment (LLACK) generated from the IEEE 802.15.4 MAC protocol. In this standard, subject (sender) kept packets in its cache until it received an LLACK from the target (receiver). When the target successfully receives the packet, it would send back an LLACK to the subject. If the subject did not receive a LLACK within a predefined threshold time, then it would retransmit that packet. Thus, after receiving the packet successfully, the target forwarded the packet to the sponsor.

The occurrence of *event (2)* was ensured by unicasting a two-hop ACK to the subject through the alternate path R-P-S as shown in Figure 2. On receiving the packet, the sponsor sent a LLACK to target as mentioned above and also a two-hop ACK to the subject through the alternate path that contained the third party (P). The alternate path was determined during the route discovery stage by exploiting the dense nature of the sensor network as discussed in Section 3.5. The subject on receiving a LLACK from the target and subsequently an acknowledgment from sponsor through the alternate path would consider that transaction as successful, else it was considered to be a failure. The number of successful and failed transactions was stored in the transaction table as described in Section 3.3.

Based on the frequency of acknowledgments sent by the sponsor, the 2-ACKT protocol could be classified as 2-ACKT with individual acknowledgment (2-ACKT-I) and 2-ACKT with group acknowledgment (2-ACKT-G). In the proposed 2-ACKT-I scheme, the sponsor sent individual acknowledgment to the subject for every data packet it received from the target through the alternate path and hence it incurred more control overhead. In order to reduce the control overhead, we also proposed a 2-ACKT-G scheme in which the individual acknowledgment was sent for first "p" packets and thereafter an acknowledgment was sent for a group of "G" data packets (group acknowledgment), where G > p. When a group acknowledgment was received within the timeout period, the number of successful transactions field was incremented by G units.

3.3. Trust Computation. In order to store the details gained during neighbor monitoring, we introduced a transaction table in the routing protocol. The observed successful and failed transactions were stored in the transaction table. The fields in the transaction table of the subject were as follows:

⟨*node id, number of successful transactions, number of failed transactions, trust level*⟩

where *node id* is the address of the neighbor, namely, target, *number of successful transactions* is incremented by 1 or G when individual or group acknowledgment is received, respectively, *number of failed transactions* is incremented by 1 when it has not received the ACK in a given timeout, and *trust level* can take an integer value from 0 to 7 as discussed in Section 3.4.

The transaction table is updated when (i) the subject receives an ACK from the sponsor through the alternate path and (ii) the subject does not receive an ACK in a given timeout period.

3.4. Trust Representation. The trust is computed based on the observed number of successful and failed transaction entries in the transaction table. The initial trust value (T_V) is assumed to be 100 since all nodes are considered as legitimate during network setup. Let T_s and T_f be the number of successful and failed transaction. Then, T_V is defined as

$$T_V = \left(\frac{T_s + \varepsilon}{T_s + T_f} \right) 100, \tag{1}$$

where ε is a constant. The computed T_V lie in the range of 0 and 100 and it is not directly stored in the transaction table as it consumes more memory. It is mapped to one of the possible trust levels (T_L) as shown in Figure 3. T_L takes an integer value which lies in the range of 0 and 7 and requires a memory

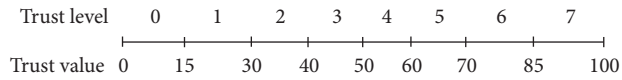

Figure 3: Trust representation.

space of 3 bits. For example, if the T_V of a target is greater than 15 and less than or equal to 30, then T_L will be 1 and stored in the transaction table. A target is considered to be trusted or untrusted based on the following condition:

$$\text{target} = \begin{cases} \text{trusted}, & T_L \geq T_{TL}, \\ \text{untrusted}, & T_L < T_{TL}, \end{cases} \quad (2)$$

where T_{TL} is the threshold trust level. It is defined as the trust level below which a node is considered to be untrusted. Any request or response from an untrusted target will not be considered during route discovery and packet forwarding stages. The threshold trust level is set by evaluating the probability of false alarm. The probability of false alarm represents the accuracy of the protocol to detect the malicious nodes. It is calculated from the statistics of the transaction table. Let n_i be the total number of neighbors (targets) for the ith node, where $i = 0, 1, 2, \ldots N-1$, where N be the total number of nodes in the network. Let l_i and m_i be the total number of legitimate and malicious nodes for which the trust assigned by the ith node falls below and lies above the threshold trust level, respectively. Then

$$\text{probability of false alarm} = \frac{1}{N}\sum_{i=0}^{N-1}\frac{l_i + m_i}{n_i}. \quad (3)$$

Figure 4 shows the probability of false alarm for different threshold trust levels. When threshold trust level is 2, the probability of false alarm is high, as malicious nodes are treated as legitimate nodes. When the threshold trust level is increased from 2, probability of false alarm decreases and reaches the minimum when the threshold trust level is 4. When the threshold trust level is further increased, probability of false alarm increases as more number of legitimate nodes is treated as malicious nodes. Hence, it is found that the optimum threshold trust level is 4 and (2) can be written as

$$\text{target} = \begin{cases} \text{trusted}, & T_L \geq 4, \\ \text{untrusted}, & T_L < 4. \end{cases} \quad (4)$$

3.5. 2-ACK Routing Protocol. When an SN (node A) desires to report an event to the BS for which a valid route is not found, it initiates a route discovery process by broadcasting a route request (RREQ) to its neighbors (node B and C) as shown in Figure 5(a). The RREQ contains the following fields:

⟨*source_address, source_seq_#, broadcast_id, destination_seq_#, hop_cnt, upstream_neighbor_address*⟩.

The pair ⟨*source_address, broadcast_id*⟩ uniquely identifies an RREQ. *broadcast_id* is incremented whenever the source

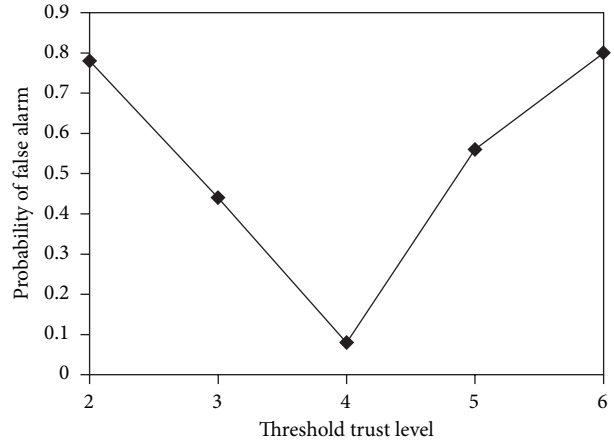

Figure 4: Probability of false alarm versus threshold trust level.

issues a new RREQ. *source_seq_#* and *destination_seq_#* is used to maintain the freshness of the route. *upstream_neighbor_address* is the address of the neighbor from which the RREQ is received. As node A initiates the RREQ, *upstream_neighbor_address* is undefined.

The algorithm for processing an RREQ packet is shown in Algorithm 1. Node B rebroadcasts the RREQ to its neighbors (node A and D) after incrementing the *hop_cnt* by one and updating the *upstream_neighbor_address* as node A as shown in Figure 5(b). In WSNs, destination address corresponds to the BS address. Each node maintains a route table that contains the following information:

⟨*destination address, next hop, number of hops, destination sequence number, active neighbors for this route, third party neighbors for this route, expiration time for the route table entry*⟩.

Associated with each route entry is a route timer which will cause the deletion of the entry if it is not used within a specified lifetime. In each route table entry, the address of active neighbors and third party neighbors are maintained. A neighbor is considered active for that destination if it originates or relays at least one packet for that destination within the most recent active_timeout period. As node B received only one copy of the RREQ packet, the third party neighbor field is undefined. Each SN maintains a route table entry for each destination of interest. Similarly node C also rebroadcasts the RREQ packet to its neighbors A and D as shown in Figure 5(c). As a result, node D receives another copy of the RREQ from A through node C with an identical *source_address, broadcast_id* and *upstream_neighbor_address*. Now node D updates the third party neighbor in the route table for the destination A as node C.

Once the RREQ reaches the BS or an intermediate SN with a fresh enough route, the BS or an intermediate SN responds by unicasting an RREP packet back to the neighbor from which it first received the RREQ. As the RREP is routed back along the reverse path, SNs along this path verify whether the RREP is received from a trusted SN and set up

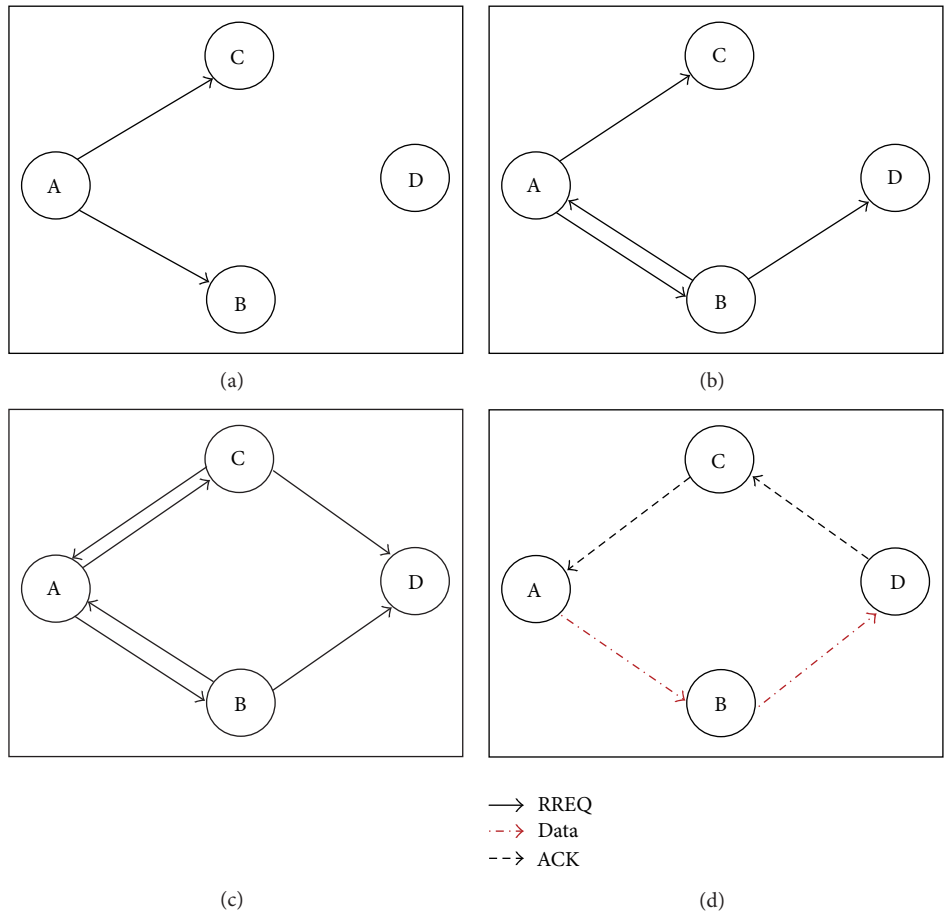

Figure 5: 2-ACK routing protocol.

forward route entries in their route tables which point to the SN from which the RREP came.

On receiving the RREP packet from node B, node A will forward the data packet to node B. The algorithm for processing the data packet is shown in Algorithm 2. Node B will forward the data packet to node D as shown in Figure 5(d). On successfully receiving the data packet from node B, node C will send an acknowledgment to the node A through the alternate path D-C-A.

4. Performance Analysis

The performances of 2-ACKT-I and 2-ACKT-G routing protocols were evaluated using ns-2 simulator. The simulation parameters are listed in Table 1. We took a simulation area of 300 m × 300 m, with six hundred nodes placed in random. The transmission range was 45 m. The IEEE 802.15.4 was the MAC layer protocol used to evaluate the performance of the proposed trust model under attack conditions.

4.1. Metrics. The performance of 2-ACKT trust model was evaluated using the following metrics.

Packet Loss. The total number of data packets lost legitimately or through malicious action without any notification.

Table 1: Simulation parameters.

Simulation time	800 Secs
Simulation area	300 m × 300 m
Number of nodes	600
Frequency of operation	2.4 GHz
Node placement	Random
Transmission range	45 m
Propagation model	Free space
Movement model	Static
Traffic type	CBR (UDP)
Packet size	50 bytes
Packet interval	10 secs
Maximum number of malicious nodes	180
Type of attack	Black hole
Initial energy	2 Joules

Packet Delivery Ratio (PDR). The ratio of total number of data packets received to the total number of data packets transmitted.

Control Overhead. The ratio of the total number of control packets generated to the total number of data packets received.

```
if (neighbor is not trusted) then
   drop the RREQ packet
else
        if (source_address & broadcast_id is not seen earlier) then
                stores the destination address, source address, broadcast_id, expiration time for reverse
                path route entry, source sequence number and upstream neighbor address in routing
                table
                if ((node address is equal to the destination address) or (node has a route to the
                destination)) then
                        send RREP packet to source
                else
                        rebroadcast RREQ after setting
                        hop_cnt = hop_cnt + 1
                        upstream_neighbor_address as the address of the upstream neighbor node
                end if
        else
                check the route table
                if (upstream_neighbor_address is not similar with the previously processed RREQ)
                then
                        drop RREQ
                else
                        update third_party_neighbor as the address of the upstream node and then drop
                        the packet
                end if
        end if
end if
```

Algorithm 1: Algorithm for processing an RREQ packet received from the neighbor.

```
if (neighbor is trusted ) then
        if (node address is equal to the destination address) then
                send a two-hop acknowledgment to the third party neighbor
        else
                forward data packet to next hop
                send a two hop acknowledgment to subject through the alternate path
        end if
else
        drop data packet
end if
```

Algorithm 2: Algorithm for processing a data packet received from the neighbor.

Energy Consumption. The average energy consumed by each node during the given simulation time and expressed in Joules (J).

Network Lifetime. Time taken for the energy of the first node to fall from 0.5 J to zero and expressed in seconds.

End-to-End Delay. The delay experienced by the data packet during transmission from source to BS, including processing, queuing and propagation delay.

Communication Overhead. The total number of packets generated for trust establishment in the network.

Memory Requirement. The total memory space exclusively used for trust derivation and representation, expressed in bits.

4.2. Simulation Results and Discussion. The performance of 2-ACKT-I and 2-ACKT-G protocols is compared with AODV [32] protocol under varying number of malicious nodes as shown in Figure 6. The malicious nodes manifest black hole attack [3] where the node drops all the received data packets instead of forwarding. We have also implemented the promiscuous mode-based trust AODV routing protocol (PMT-AODV). In PMT-AODV, node A sends a packet to node B and sets its transceiver in promiscuous mode. The transaction is considered as successful when node A receives an LLACK and a passive acknowledgment (by overhearing) from node B. If the acknowledgment is not received within a time period, the transaction is considered as failed. The numbers of successful and failed transactions are updated in the transaction table represented in Section 3.3. The trust

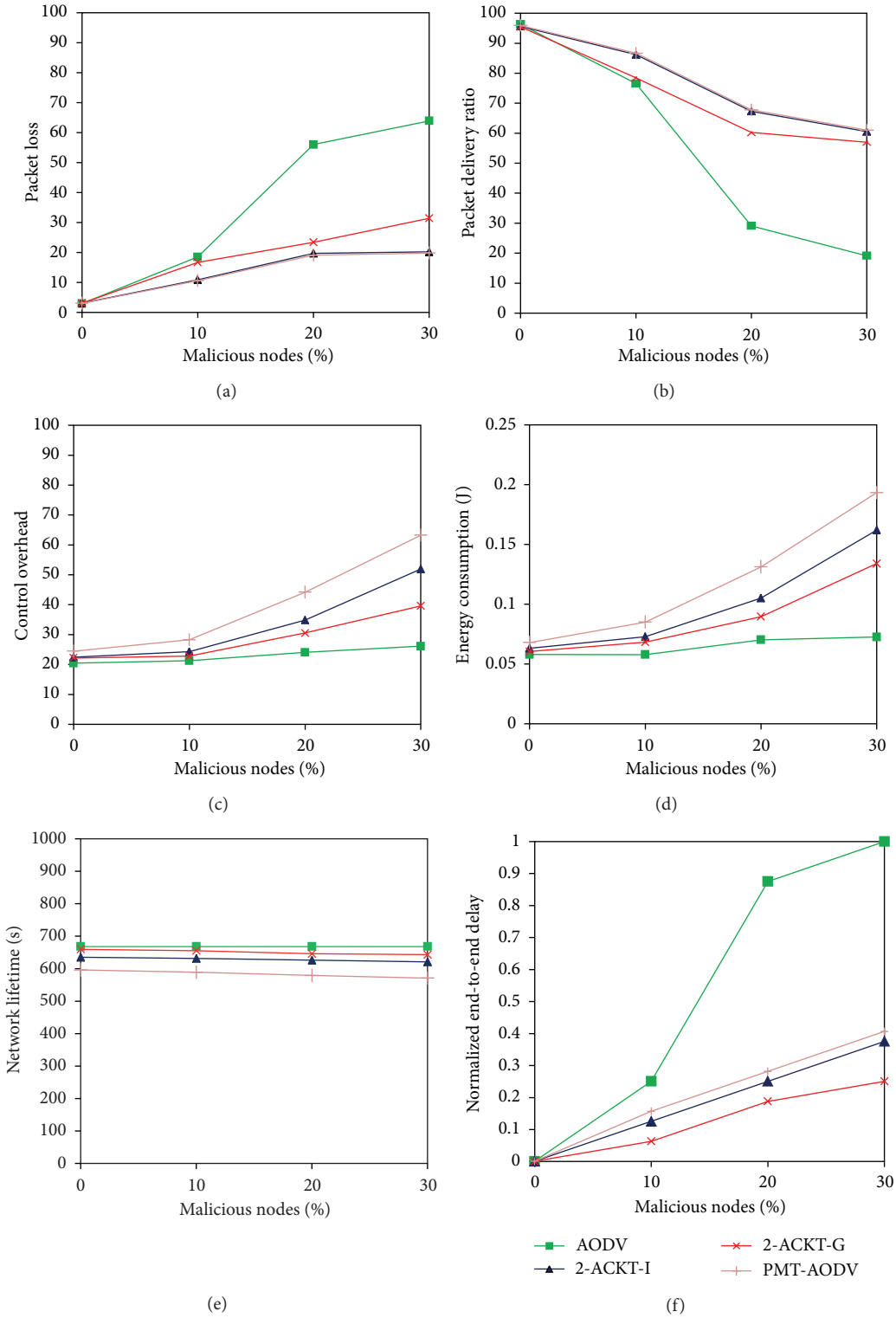

Figure 6: Performance comparison of 2-ACKT-I, 2-ACKT-G, PMT-AODV, and AODV routing protocols under varying percentage of malicious attacks.

computation and trust representation are performed as mentioned in Section 3.4. The 2-ACKT-I, 2-ACKT-G, PMT-AODV, and AODV protocols are tested against exactly the same scenario and connection pattern. The packet loss of 2-ACKT-I, 2-ACKT-G, PMT-AODV, and AODV protocols are plotted against varying percentage of malicious attacks as shown in Figure 6(a). In AODV, the discovered route to the BS may consist of malicious nodes. As a result, the packet

loss in 2-ACKT-I is 61.9 percent lower than AODV protocol. The malicious nodes are effectively identified and eliminated in the discovered route, resulting in lower packet loss in 2-ACKT-I, 2-ACKT-G, and PMT-AODV protocols. This has a positive effect on the PDR of the 2-ACKT-I, 2-ACKT-G, and PMT-AODV protocols as shown in Figure 6(b). 2-ACKT-I routing protocol augments the PDR of the AODV routing protocol by 40.12 percent. In 2-ACKT-I routing protocol, the sponsor sends ACK to the subject through the third party for every data packet received from the target. As a result, the control overhead of 2-ACKT-I is 45.44 percent higher than AODV protocol. In PMT-AODV, each SN overhears every packet forwarded by its neighbors and, as a result, the control overhead of PMT-AODV is 19.9 percent higher than 2-ACKT-I scheme. In 2-ACKT-G protocol, the sponsor sends an ACK for every group of "G" data packets received from the target. Consequently, the control overhead of 2-ACKT-G protocol is 13.81 percent lower than 2-ACKT-I protocol and 28.12 percent lower than PMT-AODV protocol as shown in Figure 6(c). The lower control overhead in 2-ACKT-G reduces energy consumption by 12.61 percent and 26.2 percent when compared to 2-ACKT-I and PMT-AODV, respectively, as shown in Figure 6(d). The simulation is performed with an initial energy of 0.5 J to calculate the network lifetime. The higher energy consumption of 2-ACKT-I, 2-ACKT-G and PMT-AODV protocols result in lower network lifetime of 5.9 percent, 2.53 percent and 12.6 percent, respectively, over AODV routing protocol as shown in Figure 6(e). The lower energy consumption of 2-ACKT-G improves the network lifetime by 3.59 percent and 11.53 percent when compared to 2-ACKT-I and PMT-AODV protocols, respectively. Even though the control overhead of AODV is lower than 2-ACKT-I, 2-ACKT-G and PMT-AODV, the presence of malicious nodes results in higher end-to-end delay in AODV as most of the packets do not reach the destination as shown in Figure 6(f).

The impact of ε on the PDR of the network was studied under 0 and 30 percent malicious attacks as shown in Table 2. When ε is low and in the absence of malicious nodes, the trust value of the legitimate neighbor falls from the trusted region ($T_L \geq 4$) to the untrusted region ($T_L < 4$) rapidly. It is due to delay in receiving the acknowledgment packet sent by the sponsor node using an unreliable wireless communication channel. As a result, the legitimate nodes are wrongly identified as malicious nodes and hence PDR is low at ε is equal to 0.5 as shown in Table 2. It can be understood that the time taken to identify the malicious node increases with ε. When ε is high, a legitimate node unknowingly forwards more number of data packets to the malicious node that performs black hole attack. Hence, the PDR falls to the minimum when $\varepsilon = 10$ in the presence of 30 percent malicious nodes. It can be seen from Table 2, the PDR of the protocol is optimum at $\varepsilon = 1$. Hence, in our simulation, ε is set at unity for maximum PDR.

Table 2: ε versus PDR.

ε	PDR	
	0% malicious nodes	30% malicious nodes
0.5	76.077	61.646
1	95.663	60.507
5	96.077	47.973
10	96.077	29.874

4.3. Theoretical Analysis. Let us assume that "N" be the total number of SNs in the network and let "h" denote the average number of hops between an SN and the BS. We have assumed that all nodes in the network want to communicate with the BS using "h" hops and do not have any prior trust relationship with their neighbors. It can be understood that when the number of interactions is less than "p", there is no difference in the operation of 2-ACKT-I and 2-ACKT-G protocols and hence they are represented simply as 2-ACKT protocol in the following analysis and discussion. In this section, the performance of 2-ACKT protocol is compared with the GTMS [26], RFSN [16], PLUS [20] and ATRM [21] protocols in terms of communication overhead and memory requirement.

4.3.1. Communication Overhead. In 2-ACKT, when a node wants to communicate with the BS in the discovered path, then the total number of acknowledgments generated for trust computation is $2(h-1)$. When all the nodes want to communicate with the BS, then the communication overhead is

$$C_{\text{2-ACKT}} = 2N(h-1). \tag{5}$$

GTMS. In GTMS [26], when a node from ith group wants to communicate with the BS through its CH, it sends a maximum of $\sigma - 2$ peer recommendation requests and in turn receives a maximum of $\sigma - 2$ peer responses, where σ is the average size of the group. Therefore, the communication overhead for the interaction between a node and its CH is

$$C_{\text{GTMS (N-CH)}} = 2(\sigma - 2). \tag{6}$$

When a CH of ith group wants to communicate with CH of jth group, it sends one peer recommendation request to the BS and receives one response from the BS. Then the communication overhead for the interaction between two adjacent CHs is $C_{\text{GTMS (CH-CH)}} = 2$. As the BS is considered to be trusted, if a node present in the ith group wants to communicate with the BS in h hops, then the total communication overhead is obtained by multiplying $C_{\text{GTMS (CH-CH)}}$ by $(h-2)$ and adding the product to (6) as given by

$$C_{\text{GTMS (N-BS)}} = 2(\sigma - 2) + 2(h-2) = 2(\sigma + h - 4). \tag{7}$$

If all the "N" number of SNs want to communicate with the BS, then the total communication overhead is obtained by multiplying (7) by N,

$$C_{\text{GTMS}} = 2N(\sigma + h - 4). \tag{8}$$

RFSN. In RFSN [16], when a node from ith group wants to communicate with the BS, it sends a maximum of $\sigma - 2$ peer recommendation requests and it receives a maximum of $\sigma - 2$ peer responses, where σ is the average size of the group. Therefore, the communication overhead for the interaction between a node and its CH is

$$C_{\text{RFSN (N-CH)}} = 2(\sigma - 2). \quad (9)$$

When a CH of ith group wants to communicate with CH of jth group, it sends $(N/\sigma - 2)$ peer recommendations to the CHs and receives $(N/\sigma - 2)$ responses, where (N/σ) is the average number of CHs. Therefore, the communication overhead for the interaction between two CHs is

$$C_{\text{RFSN (CH-CH)}} = 2\left(\frac{N}{\sigma} - 2\right). \quad (10)$$

As the BS is considered to be trusted, if a node present in the ith group wants to communicate with the BS in h hops, then the total communication overhead is obtained by multiplying (10) by $(h - 2)$ and adding the product to (9) as given by

$$C_{\text{RFSN (N-BS)}} = 2(\sigma - 2) + 2(h - 2)\left(\frac{N}{\sigma} - 2\right). \quad (11)$$

If all the "N" number of SNs want to communicate with the BS, then the total communication overhead is obtained by multiplying (11) by N,

$$C_{\text{RFSN}} = 2N\left[(\sigma - 2) + (h - 2)\left(\frac{N}{\sigma} - 2\right)\right]. \quad (12)$$

PLUS. In PLUS [20], a node broadcasts a request packet and in turn receives $\sigma - 2$ responses, for communicating with the CH. Hence, the communication overhead is

$$C_{\text{PLUS (N-CH)}} = 1 + (\sigma - 2). \quad (13)$$

For inter CH communication, the CH broadcasts a request packet and receives a maximum of $(N/\sigma - 2)$ responses. Therefore, the communication overhead is

$$C_{\text{PLUS (CH-CH)}} = 1 + \left(\frac{N}{\sigma} - 2\right). \quad (14)$$

As the BS is considered to be trusted, if a node present in the ith group wants to communicate with the BS in h hops, then the total communication overhead is obtained by multiplying (14) by $(h - 2)$ and adding the product to (13) as given by

$$C_{\text{PLUS (N-BS)}} = 1 + (\sigma - 2) + (h - 2)\left[1 + \left(\frac{N}{\sigma} - 2\right)\right]. \quad (15)$$

If all the "N" number of SNs want to communicate with the BS, then the total communication overhead is obtained by multiplying (15) by N,

$$C_{\text{PLUS}} = N\left\{1 + (\sigma - 2) + (h - 2)\left[1 + \left(\frac{N}{\sigma} - 2\right)\right]\right\}. \quad (16)$$

ATRM. In ATRM [21], a node transmits four packets to communicate with the CH and hence the communication overhead is

$$C_{\text{ATRM (N-CH)}} = 4. \quad (17)$$

When a CH of ith group wants to communicate with CH of jth group, it transmits four packets. Then, the communication overhead is

$$C_{\text{ATRM (CH-CH)}} = 4. \quad (18)$$

As the BS is considered to be trusted, if a node present in the ith group wants to communicate with the BS in h hops, then the total communication overhead is obtained by multiplying (18) by $(h - 2)$ and adding the product to (17) as given by

$$C_{\text{ATRM (N-BS)}} = 4[1 + (h - 2)]. \quad (19)$$

If all the "N" number of SNs want to communicate with the BS, then the total communication overhead is obtained by multiplying (19) by N,

$$C_{\text{ATRM}} = 4N[1 + (h - 2)]. \quad (20)$$

The communication overheads of 2-ACKT, GTMS, RFSN, ATRM, and PLUS protocols were studied by setting $N = 144$ and $h = 5$ under varying number of communicating nodes as shown in Figure 7(a). In the graph, the GTMS with a cluster size of 9, 12, and 18 are represented as GTMS-9, GTMS-12, and GTMS-18, respectively. A similar convention is followed to represent RFSN and PLUS protocols with a cluster size of 9, 12, and 18. It can be seen that the use of two-hop acknowledgment for trust derivation significantly reduces the communication overhead of 2-ACKT protocol. In RFSN-9, each node transmits peer recommendation request to all the neighboring nodes and receives peer responses. As a result, the communication overhead of RFSN-9 is 89 percent higher than 2-ACKT protocol as shown in Figure 7(a). The communication overhead is slightly reduced in PLUS-9 as each node broadcasts only one recommendation request packet and receives peer responses from the neighbors. In ATRM, each node exchanges four packets for trust derivation instead of peer recommendation request and response. As a result, the communication overhead of ATRM-9 is 30 percent lower than PLUS-9 as shown in Figure 7(a). When the cluster size decreases, the communication overhead increases both in RFSN and PLUS protocols. In GTMS, the communication overhead decreases with the cluster size and it is almost equal to 2-ACKT when $\sigma = 1$. But, GTMS uses a high energy CH which has the ability to directly send a request to the BS and gather a recommendation about the state of the neighboring CHs. As a result, the GTMS scheme is not suitable for homogeneous WSNs.

It can be understood that when the number of interactions increases, the communication overhead increases in 2-ACKT-I by the factor 2 per interaction and thus making it suitable for low traffic sensor network. However, the group acknowledgment used in 2-ACKT-G does not increase the communication overhead linearly with the interactions and thus making it suitable for low as well as high data traffic sensor networks.

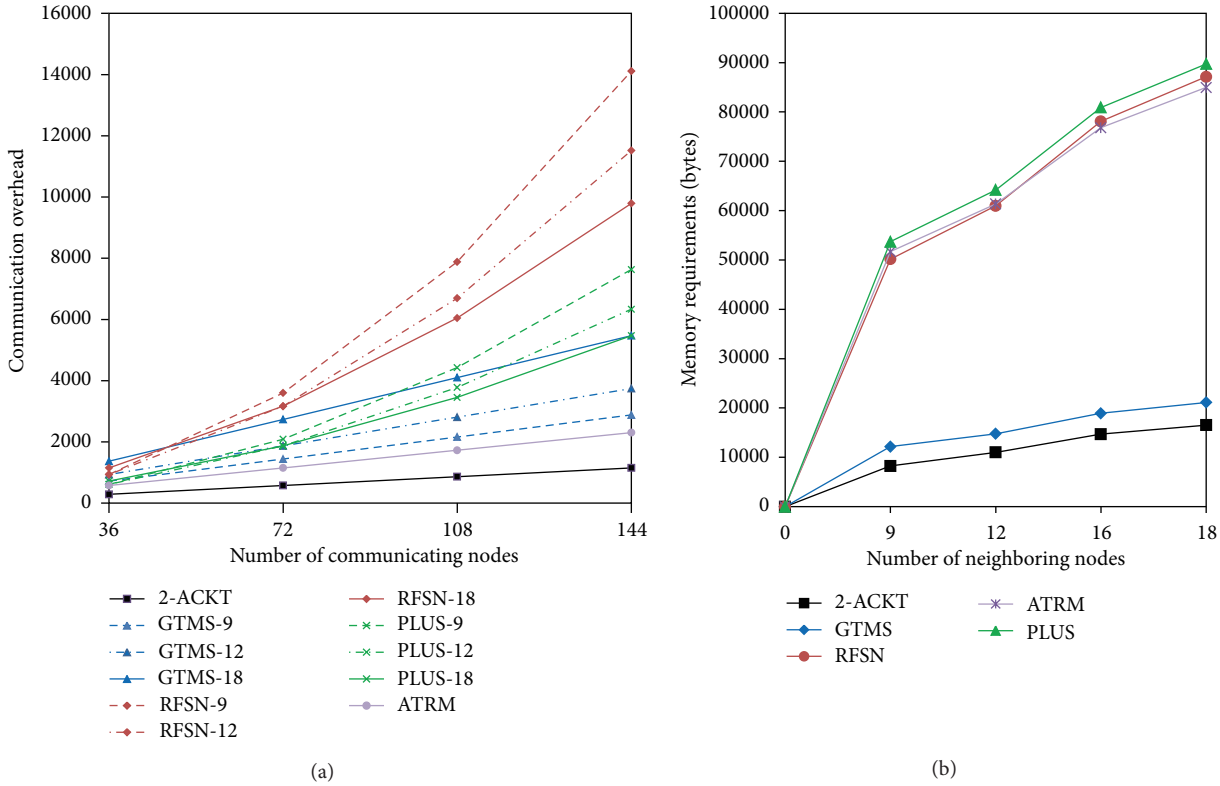

Figure 7: Performance comparison of 2-ACKT, GTMS, RFSN, ATRM, and PLUS protocols.

Table 3: 2-ACKT transaction table.

Node id	Number of successful transactions	Number of failed transactions	Trust level
2 bytes	2 bytes	2 bytes	3 bits

4.3.2. Memory Consumption. In 2-ACKT, SN maintains a transaction table to monitor and store the trust level of their neighbors. The fields in the transaction table and its memory size are shown in the Table 3. The node id, number of successful transactions and number of failed transactions occupies 2 bytes each, and trust level requires 3 bits. Therefore, the memory required to store a record in the transaction table that represents the trust relationship with a neighbor is 6.375 bytes. The total size of the transaction table that represents the trust relationship between an SN and all its neighbors is

$$M_{2\text{-ACK (SN)}} = 6.375n \text{ bytes}, \quad (21)$$

where n is the average number of neighbors for each SN.

Therefore, the total memory requirement for the entire network which consists of N number of sensors is

$$M_{2\text{-ACKT}} = 6.375nN \text{ bytes}. \quad (22)$$

GTMS. In GTMS [26], each SN maintains a trust table to store the intra group trust relationship with its neighbors. In addition to this, each CH maintains a group trust table to store the trust relationship with the other CHs in the network. The memory size of the each record in the trust and group trust table is largely determined by the size of the time window (Δt) used to derive the number of successful and unsuccessful transactions. Assuming the minimum possible value for Δt as unity, the memory size for the fields used in the GTMS trust table is shown in Table 4. The node id, number of successful transactions and number of unsuccessful transactions fields occupy 2 bytes each; peer recommendation and trust value fields require 1 byte each. The memory requirement of each record maintained in the trust table is 8 bytes. Therefore, the size of the trust table to store the intra group trust relationship with the neighbors is

$$M_{\text{GTMS (SN)}} = 8(\sigma - 1) \text{ bytes}, \quad (23)$$

where σ is the average number of nodes in the group.

Table 4: GTMS trust table.

Node id	Number of successful transactions	Number of unsuccessful transactions	Peer recommendation	Trust value
2 bytes	2 bytes	2 bytes	1 byte	1 byte

Table 5: GTMS group trust table.

CH node id	Number of successful transactions	Number of unsuccessful transactions	Peer recommendation	Trust value
2 bytes	2 bytes	2 bytes	1 byte	1 byte

Table 6: RFSN reputation module matrix.

Context	Aging period	Aging weight	Integration weight	Size	Alpha	Beta	Node id
4 bytes	4 bytes	4 bytes	4 bytes	1 bytes	4 bytes	4 bytes	2 bytes

Table 7: RFSN monitor.

Node id	Data readings
2 bytes	4 bytes

Assuming the minimum possible value for Δt as unity, the size of each record that represents the trust relationship with other CHs in the GTMS group trust table is also is 8 bytes long as shown in Table 5. The memory size required to store the group trust table in a CH is

$$M_{\mathrm{GTMS\,(CH)}} = 8\left(\frac{N}{\sigma} - 1\right) + 8(\sigma - 1) = 8\left(\frac{N}{\sigma} + \sigma - 2\right)\text{bytes}, \quad (24)$$

where (N/σ) represents the number of CHs in the network.

The total memory required for a network that consists of N number of nodes and (N/σ) number of CHs is determined by adding the products obtained by multiplying (23) by N and (24) by (N/σ) as given by

$$M_{\mathrm{GTMS}} = 8N(\sigma - 1) + 8\left(\frac{N}{\sigma}\right)\left(\frac{N}{\sigma} + \sigma - 2\right). \quad (25)$$

Assuming $\sigma = n$, (25) is rewritten as

$$M_{\mathrm{GTMS}} = 8N(n - 1) + 8\left(\frac{N}{n}\right)\left(\frac{N}{n} + n - 2\right). \quad (26)$$

RFSN. In RFSN [16], each SN and CH maintains two tables, that is, Reputation Module Matrix (RMM) of 27 bytes and RFSN monitor of 6 bytes. The memory size for the fields used in the RFSN tables is shown in Tables 6 and 7. The memory requirement of each record maintained in the RMM table and monitor is 33 bytes. Therefore, the size of the trust tables to store the trust relationship with the neighbors is

$$M_{\mathrm{RFSN\,(SN)}} = 33(\sigma - 1)\,\text{bytes}, \quad (27)$$

where σ is the average number of nodes in the group.

The memory size required to store the trust tables in a CH is

$$M_{\mathrm{RFSN\,(CH)}} = 33\left(\frac{N}{\sigma} - 1\right) + 33(\sigma - 1) = 33\left(\frac{N}{\sigma} + \sigma - 2\right)\text{bytes}, \quad (28)$$

where (N/σ) represents the number of CHs in the network.

The total memory required for a network that consists of N number of nodes and (N/σ) number of CHs is determined by adding the products obtained by multiplying (27) by N and (28) by (N/σ) as given by

$$M_{\mathrm{RFSN}} = 33N(\sigma - 1) + 33\left(\frac{N}{\sigma}\right)\left(\frac{N}{\sigma} + \sigma - 2\right). \quad (29)$$

Table 8: ATRM trust evaluation table.

Node id	Trust context	Evaluation	Time stamp
2 bytes	4 bytes	4 bytes	4 bytes

Table 9: ATRM t_Instrument table.

Node id	Trust context	INSTR	Time stamp	ACK
2 bytes	4 bytes	4 bytes	4 bytes	2 bytes

Assuming $\sigma = n$, (29) is rewritten as

$$M_{\mathrm{RFSN}} = 33N(n - 1) + 33\left(\frac{N}{n}\right)\left(\frac{N}{n} + n - 2\right). \quad (30)$$

ATRM. In ATRM [21], each SN and CH maintains two tables, that is, trust evaluation table of 14 bytes and t_Instrument table of 16 bytes for each record. The SN and CH also maintain r_certificate and the memory size of r_certificate are largely determined by the number of available context and the hash function used. Each context requires 8 bytes and the other fields such as node id (2 bytes), time stamp (4 bytes). Assume that the hash function used is MD5 (16 bytes). Therefore, the r_certificate requires $22 + 8k$ bytes, where "k" represents the number of context. The memory size for the fields used in the ATRM tables is shown in Tables 8 and 9. The memory requirement of each record maintained in the ATRM tables and r_certificate is

$$M_{\mathrm{ATRM\,(SN)}} = 30 + (22 + 8k)\,\text{bytes}. \quad (31)$$

Therefore, the size of the trust tables to store the trust relationship with the neighbors is

$$M_{\mathrm{ATRM\,(SN)}} = 30(\sigma - 1) + 22 + 8k = 30\sigma + 8(k - 1)\,\text{bytes}, \quad (32)$$

where σ is the average number of nodes in the group.

The memory size required to store the trust tables in a CH is

$$M_{\mathrm{ATRM\,(CH)}} = 30\left(\frac{N}{\sigma} - 1\right) + 30(\sigma - 1) + 22 + 8k = 30\left(\frac{N}{\sigma} + \sigma\right) + 2(4k - 19)\,\text{bytes}, \quad (33)$$

where (N/σ) represents the number of CHs in the network.

Table 10: PLUS trust estimator table.

Node id	Personal reference parameter						Peer recommendation value	Trust value
	T_{or}	T_{ai}	T_{cp}	T_{co}	T_{re}	T_{po}		
2 bytes	1 bit	1 bit	1 bit	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

Table 11: PLUS network I/O table.

Node id	Number of requests sent	Number of replies received	Number of packets forwarded	Number of packets to be forwarded
2 bytes	2 bytes	2 bytes	2 bytes	2 bytes

The total memory required for a network that consists of N number of nodes and (N/σ) number of CHs is determined by adding the products obtained by multiplying (32) by N and (33) by (N/σ) as given by

$$M_{\text{ATRM}} = N\,(30\sigma + 8\,(k-1)) + \left(\frac{N}{\sigma}\right)\left[30\left(\frac{N}{\sigma}+\sigma\right) + 2\,(4k-19)\right] \text{ bytes.} \tag{34}$$

Assuming $\sigma = n$, (34) is rewritten as

$$M_{\text{ATRM}} = N\,(30n + 8\,(k-1)) + \left(\frac{N}{n}\right)\left[30\left(\frac{N}{n}+n\right) + 2\,(4k-19)\right] \text{ bytes.} \tag{35}$$

PLUS. In PLUS [20], each SN and CH maintains trust estimator table of 22.375 bytes and network I/O table of 10 bytes for each record. It also maintains seven context parameters and requires 4 bytes for each parameter. The memory size required for the fields used in the PLUS tables is shown in Tables 10 and 11. The size of the two tables and context parameters in SN to store the trust relationship with the neighbors is

$$\begin{aligned} M_{\text{PLUS (SN)}} &= 22.375\,(\sigma-1) + 10\,(\sigma-1) + 28 \\ &= 32.375\,(\sigma-1) + 28 \text{ bytes,} \end{aligned} \tag{36}$$

where σ is the average number of nodes in the group.

The memory size required to store the trust tables in a CH is

$$\begin{aligned} M_{\text{PLUS (CH)}} &= 32.375\left(\frac{N}{\sigma}-1\right) + 32.375\,(\sigma-1) + 28 \\ &= 32.375\left(\frac{N}{\sigma}+\sigma-2\right) + 28 \text{ bytes,} \end{aligned} \tag{37}$$

where (N/σ) represents the number of CHs in the network.

The total memory required for a network that consists of N number of nodes and (N/σ) number of CHs is determined by adding the products obtained by multiplying (36) by N and (37) by (N/σ) as given by

$$M_{\text{PLUS}} = N\,(32.375\,(\sigma-1) + 28) + \left(\frac{N}{\sigma}\right)\left[32.375\left(\frac{N}{\sigma}+\sigma-2\right) + 28\right] \text{ bytes.} \tag{38}$$

Assuming $\sigma = n$, (38) is rewritten as

$$M_{\text{PLUS}} = N\,(32.375\,(n-1) + 28) + \left(\frac{N}{n}\right)\left[32.375\left(\frac{N}{n}+n-2\right) + 28\right] \text{ bytes.} \tag{39}$$

The memory requirement of 2-ACKT, GTMS, RFSN, ATRM, and PLUS schemes was studied by varying the number of neighbors for an SN in the network as shown in Figure 7(b). The memory required for 2-ACKT scheme is 24.64 percent lower than the GTMS scheme. RFSN, ATRM, and PLUS schemes require nearly 82 percent more memory than 2-ACKT routing protocol. The higher memory requirement of the above compared schemes is mainly due to the additional tables used to support trust computation.

5. Conclusions

Security is an important problem that can significantly degrade the performance of resource constrained WSNs. In this research work, we have proposed a new 2-ACKT framework to thwart malicious attacks in WSNs. It relies on a two-hop acknowledgment from the sponsor to derive the trust on neighboring nodes. This scheme exploits the dense nature of the sensor networks to establish trust in the network. The simulation and theoretical results show that the proposed 2-ACKT protocol has better performance than the conventional multihop and trust-based routing protocols in terms of packet delivery ratio, network lifetime, end-to-end delay, communication overhead, and memory consumption making it suitable for homogeneous WSNs. In this research work, the malicious attacks are manifested by individual sensor nodes. However, there exists a much wider spectrum of security threats involving collaborative attackers. Hence, we plan to design a comprehensive trust-based security solution that thwarts collaborative attackers in a resource constrained WSNs.

References

[1] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: a survey," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 2, pp. 52–73, 2009.

[2] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proceedings of IEEE Interntional Conference on Acoustics, Speech, and Signal Processing*, pp. 2033–2036, Salt Lake City, Utah, USA, May 2001.

[3] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communication Surveys and Tutorials*, vol. 8, no. 2, pp. 2–23, 2006.

[4] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Computing Surveys*, vol. 42, no. 1, article 1, pp. 1–31, 2009.

[5] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[6] J. Lopez, R. Roman, and C. Alcaraz, "Analysis of security threats, requirements, technologies and standards in wireless sensor networks," in *Foundations of Security Analysis and Design*, pp. 289–338, 2009.

[7] B. Xiao and B. Yu, "Detecting selective forwarding attacks in wireless sensor networks," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS '06)*, pp. 1–8, 2006.

[8] Y. Yu, K. Li, W. Zhou, and P. Li, "Trust mechanisms in wireless sensor networks: attack analysis and countermeasures," *Journal of Network and Computer Applications*, vol. 35, pp. 867–880, 2012.

[9] I. K. Maarouf and A. R. Naseer, "WSNodeRater—An optimized reputation system framework for security aware energy efficient geographic routing in WSNs," in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications (AICCSA '07)*, pp. 258–265, May 2007.

[10] E. Aivaloglou, S. Gritzalis, and C. Skianis, "Trust establishment in sensor networks: behaviour-based, certificate-based and a combinational approach," *International Journal of System of Systems Engineering*, vol. 1, no. 1-2, pp. 128–148, 2008.

[11] M. Momani, S. Challa, and R. Alhmouz, "Can we trust trusted nodes in wireless sensor networks?" in *Proceedings of the International Conference on Computer and Communication Engineering (ICCCE '08)*, pp. 1227–1232, May 2008.

[12] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy, "Location-centric isolation of misbehavior and trust routing in energy-constrained sensor networks," in *Proceedings of IEEE Workshop on Energy-Efficient Wireless Communications and Networks (EWCN '04), in conjunction with IEEE International Conference on Performance, Computing and Communications (IPCCC '04)*, Phoenix, Ariz, USA, April 2004.

[13] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[14] A. Srinivasan and J. Wu, "A novel k-Parent flooding tree for secure and reliable broadcasting in sensor networks," in *Proceedings of IEEE International Conference on Communications (ICC '07)*, pp. 1497–1502, June 2007.

[15] A. Srinivasan and J. Wu, "Secure and reliable broadcasting in wireless sensor networks using multi-parent trees," *Security and Communication Networks*, vol. 2, no. 3, pp. 239–253, 2009.

[16] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," in *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 66–77, October 2004.

[17] Y. Yang, Q. Feng, Y. L. Sun, and Y. Dai, "RepTrap: A novel attack on feedback-based reputation systems," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, Istanbul, Turkey, September 2008.

[18] C. Doekjai, H. Guandjie, and L. Wontaek, "A reliable approach of establishing trust for wireless sensor networks," in *Proceedings of the IFIP International Conference on Network and Parallel Computing Workshops (NPC '07)*, pp. 232–237, chn, September 2007.

[19] S. Ozdemir, "Functional reputation based reliable data aggregation and transmission for wireless sensor networks," *Computer Communications*, vol. 31, no. 17, pp. 3941–3953, 2008.

[20] Z. Yao, D. Kim, and Y. Doh, "PLUS: Parameterized and localized trust management scheme for sensor networks security," in *Proceedings of the 3rd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS '06)*, pp. 437–446, October 2006.

[21] A. Boukerch, X. Li, and K. EL-Khatib, "Trust-based security for wireless ad hoc and sensor networks," *Computer Communications*, vol. 30, no. 11-12, pp. 2413–2427, 2007.

[22] J. Hu, H. Chen, C. Gao, and W. Huafeng, "Agent-based trust management model for wireless sensor networks," in *Proceedings of the 2nd International Conference on Multimedia and Ubiquitous Engineering (MUE '08)*, pp. 150–154, April 2008.

[23] F. Li, A. Srinivasan, and J. Wu, "A novel CDS-based reputation monitoring system for wireless sensor networks," in *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops (ICDCS '08)*, pp. 364–369, June 2008.

[24] I. Maarouf, U. Baroudi, and A. R. Naseer, "Efficient monitoring approach for reputation system-based trust-aware routing in wireless sensor networks," *IET Communications*, vol. 3, no. 5, pp. 846–858, 2009.

[25] A. Rezgui and M. Eltoweissy, "μRACER: A reliable adaptive service-driven efficient routing protocol suite for sensor-actuator networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 5, pp. 607–622, 2009.

[26] R. A. Shaikh, H. Jameel, B. J. d'Auriol, H. Lee, S. Lee, and Y. J. Song, "Group-based trust management scheme for clustered wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 11, pp. 1698–1712, 2009.

[27] J. Zhang, R. Shankaran, M. A. Orgun, V. Varadharajan, and A. Sattar, "A trust management architecture for hierarchical wireless sensor networks," in *Proceedings of the 35th Annual IEEE Conference on Local Computer Networks (LCN '10)*, pp. 264–267, October 2010.

[28] H. A. Rahhal, I. A. Ali, and S. I. Shaheen, "A novel trust-based cross-layer model for wireless sensor networks," in *Proceedings of the 28th National Radio Science Conference (NRSC '11)*, pp. 1–10, April 2011.

[29] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, October 2003.

[30] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 197–213, May 2003.

[31] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 162–175, November 2004.

[32] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pp. 90–100, February 1999.