TARF: A Trust-Aware Routing Framework for Wireless Sensor Networks*

Guoxing Zhan[1], Weisong Shi[1], and Julia Deng[2]

[1] Wayne State University, 5143 Cass Avenue, Detroit, MI 48202, USA
{gxzhan,weisong}@wayne.edu
[2] Intelligent Automation Inc., 15400 Calhoun, Rockville, MD 20855, USA
hdeng@i-a-i.com

Abstract. Multi-hop routing in wireless sensor networks (WSNs) offers little protection against deception through replaying routing information. This defect can be taken advantage of by an adversary to misdirect significant network traffic, resulting in disastrous consequences. It cannot be solved solely by encryption or authentication techniques. To secure multi-hop routing in WSNs against intruders exploiting the replay of routing information, we propose TARF, a trust-aware routing framework for WSNs. Not only does TARF significantly reduce negative impacts from these attackers, it is also energy-efficient with acceptable overhead. It incorporates the trustworthiness of nodes into routing decisions and allows a node to circumvent an adversary misdirecting considerable traffic with a forged identity attained through replaying. Both our empirical and simulated experimental results indicate that TARF satisfactorily performs routing and is resilient against attacks by exploiting the replay of routing information.

1 Introduction

Wireless sensor networks (WSNs) are ideal candidates for applications such as military surveillance and forest fire monitoring to report detected events of interest. With a narrow radio communication range, a sensor node wirelessly sends messages to a base station via a multi-hop path. However, the multi-hop routing of WSNs often becomes the target of malicious attacks. In such an attack, the attacker may tamper nodes physically, create traffic collision with seemingly valid transmission, drop or misdirect messages in routes, or jam the communication channel by creating radio interference [18]. This paper focuses on the kind of attack in which an adversary misdirects packets by identity deception through replaying routing information. With such identity deception, the adversary is capable of launching harmful and hard-to-detect attacks to misdirect traffic, such as selective forwarding as well as wormhole and sinkhole attacks [8].

As an effective and easy-to-implement type of attack, a malicious node simply replays all the routing information sent from another valid node to forge the latter node's identity, thus misdirecting the network traffic. Those packets, including their original headers, are replayed without any modification. Even if this malicious node cannot directly overhear the valid node's wireless transmission, it can collude with other malicious nodes to receive those routing packets and replay them somewhere far away from

* This work is supported in part by NSF grant CNS-0721456.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 65–80, 2010.

the original valid node, which is known as a wormhole attack. Since a node in a WSN usually relies solely on the packets received to know about the sender's identity, replaying routing packets allows the malicious node to forge the identity of this valid node. After "stealing" that valid identity, this malicious node is able to misdirect the network traffic. In a selective forwarding attack, it may drop packets received, forward packets to another node not supposed to be in the routing path, or even form a transmission loop through which packets are passed among a few malicious nodes infinitely. It is often difficult to know whether a node forwards received packets correctly even with overhearing techniques [8]. Sinkhole attacks are another kind of attacks that can be launched after stealing a valid identity. In a sinkhole attack, a malicious node may claim itself to be a base station through replaying all the packets from a real base station. Such a fake base station could lure more than half the traffic, creating a "black hole".

Unfortunately, most existing routing protocols for WSNs either focus on energy efficiency [1] assuming that each node is honest with its identity, or they try to exclude unauthorized participation by encrypting data and authenticating packets. Examples of these encryption and authentication schemes for WSNs include TinySec [7], Spins [14], TinyPK [16], and TinyECC [10]. Admittedly, it is important to consider efficient energy usage for battery-powered sensor nodes and the robustness of routing under topological changes and common faults in a wild environment. However, it is also significant to incorporate security as one of the most important goals; meanwhile, even with perfect encryption and authentication, by replaying routing information, a malicious node can still participate in the network using another valid node's identity.

In contrast, trust management [2] has been introduced into peer-to-peer networks and general ad hoc networks to support decision-making [6,15], improve security [3,11], and promote node collaboration [5] and resource sharing [9]. Basically, trust management assigns each node a trust value according to its past performance. These studies target general ad hoc networks and peer-to-peer networks but not resource-constrained WSNs. Additionally, they do not address attacks arising from the replay of routing information. With a similar idea, S. Ganeriwal, L. Balzano, and M. Srivastava also proposed a reputation-based approach to detect uncooperative nodes in WSNs [4]; however, they do not address the attacks by exploiting the replay of routing information. The authors also studied the trustworthiness of the data collected by WSNs [19].

At this point, to fight against the "identity theft" threat arising from packet replaying, we introduce trust management into WSNs, proposing TARF - a trust-aware routing framework for wireless sensor networks. TARF identifies those malicious nodes that misuse "stolen" identities to misdirect packets by their low trustworthiness, thus helping nodes circumvent those attackers in their routing paths. We present the assumptions and goals of this work in Section 2, the detailed design of TARF in Section 3, our implementation of TARF in Section 4 and simulation results in Section 5. Finally, we conclude this work in Section 6.

2 Assumptions and Goals

We target secure routing for data collection tasks, which are one of the most fundamental functions of WSNs. In a data collection task, sensor nodes send sampled data to a

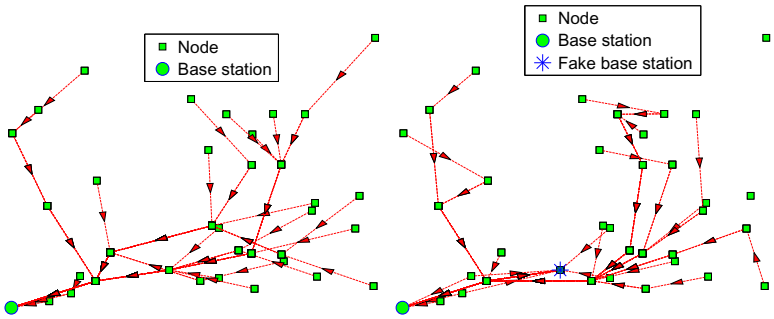


Fig. 1. Multi-hop routing: (a) normal scenarios; (b) a fake base station attracts traffic

remote base station with the aid of intermediate nodes, as in Figure 1(a). It is possible for an adversary to replay all the packets from a base station and thus to forge the identity of the base station. Such deception could result in the following situation: a large amount of packets are attracted to this fake base station and are never delivered to the real base station (see Figure 1(b)).

Though there could be more than one base station, our routing approach is not affected by the number of base stations; to simplify our discussion, we will assume that there is only one base station. Further, we assume no data aggregation is involved. Nonetheless, our approach can still be applied to static-cluster-based WSNs, where data are aggregated by static clusters before being relayed. In a static-cluster-based WSN, cluster headers themselves form a sub-network; after certain data reach a cluster header, the aggregated data will be routed to a base station only through such a sub-network consisting of cluster headers. Our framework can then be applied to this sub-network to achieve secure routing for static-cluster-based WSNs.

Additionally, we make certain assumptions regarding the format of packets in TARF. We assume all data packets and routing packets, including their packet headers, are authenticated; a packet can be forwarded only after its authenticity is verified. Whether data encryption is implemented can be decided by the application. Every data packet is assumed to have at least the following fields: the sender id, the sender sequence number, the next-hop node id (the receiver in this one-hop transmission), the source id (the node that initiates the data), and the source's sequence number. We insist that the source node's information should be included for the following reasons. First, that allows the base station to identify which data packets are initiated but undelivered; Second, a WSN cannot afford the overhead to transmit all the one-hop information to the base station. Regarding routing packets, they should have at least the following fields: the source id, the source's sequence number, and the next-hop id. In addition, we assume that after receiving a data packet, a node will send out an acknowledgement packet.

Next, we present the goals of TARF.

High Throughput: *Throughput* is defined as the ratio of the number of data packets delivered to the base station to the number of all sampled data packets. Note that single-hop re-transmission may happen, and that identical packets repeatedly transmitted are

considered as one packet as far as *throughput* is concerned. Instead of any specific data, users usually care much more about *throughput*. Here we regard high *throughput* as one of our most important goals.

Energy Efficiency: Efficient energy usage is significant for battery-powered sensor nodes, and data transmission accounts for a major portion of energy consumption. We evaluate energy efficiency by the average energy cost to successfully deliver a unit-sized data packet from a source node to the base station. Note that link-level re-transmission should be given enough attention when considering energy cost since each re-transmission causes a noticeable increase in energy consumption. If every node in a WSN consumes approximately the same energy to transmit a unit-sized data packet, we can use another metric *hop-per-delivery* to evaluate energy efficiency. Under that assumption, the energy consumption depends on the number of hops, i.e. the number of one-hop transmissions occurring. To evaluate how efficiently energy is used, we can measure the average hops per delivery, i.e., the number of all hops divided by the number of all delivered data packets, abbreviated as *hop-per-delivery*.

Excellent Scalability & Adaptability: TARF should work well with WSNs of large magnitude under highly dynamic contexts.

Here we do not include other aspects such as latency, load balance, or fairness. Low latency, balanced network load, and good fairness requirements can be enforced in specific routing protocols built on top of TARF.

3 Design of TARF

TARF secures the multi-hop routing in WSNs against intruders exploiting the replay of routing information by evaluating the trustworthiness of neighboring nodes. It identifies such intruders that misdirect noticeable network traffic by their low trustworthiness and routes data through paths circumventing those intruders to achieve satisfactory *throughput*. TARF is also energy-efficient, highly scalable, and well adaptable. Before introducing the detailed design, we first introduce several necessary notions here.

Neighbor: For a node N, a neighbor (neighboring node) of N is a node that is reachable from N with one-hop wireless transmission.

Trust level: For a node N, the trust level of a neighbor is a decimal number in $[0, 1]$, representing N's opinion of that neighbor's level of trustworthiness. Specifically, the trust level of the neighbor is N's estimation of the probability that this neighbor correctly delivers data received to the base station. That trust level is denoted as T in this paper.

Energy cost: For a node N, the energy cost of a neighbor is the average energy cost to successfully deliver a unit-sized data packet with this neighbor as its next-hop node, from N to the base station. That energy cost is denoted as E in this paper.

3.1 Overview

TARF integrates trustworthiness and energy efficiency in making routing decisions. For a node N to route a data packet to the base station, N only needs to decide to which

neighboring node it should forward the data packet. That chosen neighbor is N's next-hop node. Once the data packet is forwarded to that next-hop node, the remaining task to deliver the data to the base station is fully delegated to it, and N is totally unaware of what routing decision its next-hop node makes. To choose its next-hop node, N considers both the trustworthiness and the energy efficiency of its neighbors. For that, N maintains a neighborhood table with trust level values and energy cost values for certain known neighbors. It is sometimes necessary to delete some neighbors' entries to keep the table size acceptable. Maintaining a neighborhood table with acceptable overhead proved possible in [17]; the same technique can be used by TARF.

In TARF, in addition to data packet transmission, there are two types of routing information that need to be exchanged: broadcast messages from the base station about undelivered data packets and energy cost report messages from each node. Neither message needs acknowledgement. A broadcast message from the base station is broadcast to the whole network; each node receiving a fresh broadcast message from the base station will broadcast it to all its neighbors once. The freshness of a broadcast message is ensured by its field of source sequence number. The other type of exchanged routing information is the energy cost report message from each node, which is broadcast to only its neighbors once. Additionally, any node receiving such an energy cost report message will not forward it.

For each node N in a WSN, to maintain such a neighborhood table with trust level values and energy cost values for certain known neighbors, two components, *EnergyWatcher* and *TrustManager*, run on the node (Figure 2). *EnergyWatcher* is responsible for recording the energy cost for each known neighbor, based on N's observation of one-hop transmission to reach its neighbors and the energy cost report from those neighbors. *TrustManager* is responsible for tracking trust level values of neighbors based on network loop discovery and broadcast messages from the base station about undelivered data packets. Once N is able to decide its next-hop neighbor according to its neighborhood table, it sends out its energy report message: it broadcasts to all its neighbors its energy cost to deliver a packet from the node to the base station. The energy cost is computed as in Section 3.3 by *EnergyWatcher*. Such an energy cost report also serves as the input of its receivers' *EnergyWatcher*.

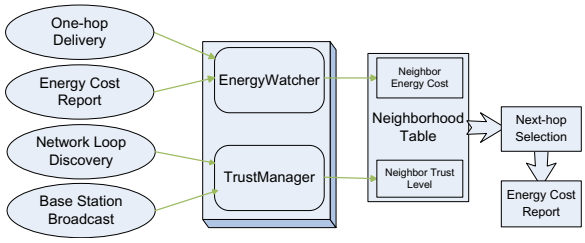


Fig. 2. Each node selects a next-hop node based on its neighborhood table, and broadcast its energy cost within its neighborhood. To maintain this neighborhood table, *EnergyWatcher* and *TrustManager* on the node keep track of related events (on the left) to record the energy cost and the trust level values of its neighbors.

3.2 Routing Procedure

TARF, as with many other routing protocols, runs as a periodic service. The length of that period determines how frequently routing information is exchanged and updated. At the beginning of each period, the base station broadcasts the information about undelivered data packets during the past few periods to the whole network once, which triggers the exchange of routing information in this new period. Whenever a node receives such a broadcast message from the base station, it knows that the most recent period has ended and a new period has just started. In this way, no time synchronization is required for a node to keep track of the beginning or ending of a period. During each period, the *EnergyWatcher* on a node monitors energy consumption of one-hop transmission to its neighbors and processes energy cost reports from those neighbors to maintain energy cost entries in its neighborhood table; its *TrustManager* also keeps track of network loops and processes broadcast messages from the base station about undelivered data to maintain trust level entries in its neighborhood table.

To maintain the stability of its routing path, a node may retain the same next-hop node until the next fresh broadcast message from the base station occurs. Meanwhile, to reduce traffic, its energy cost report could be configured to not occur again until the next fresh broadcast from the base station. If a node does not change its next-hop node selection until the next broadcast from the base station, that guarantees all paths to be loop-free, as can be deducted from the procedure of next-hop node selection. However, as noted in our experiments, that would lead to slow improvement in routing paths. Therefore, we allow a node to change its next-hop selection in a period only when its current next-hop is not responding correctly.

Next, we introduce the structure and exchange of routing information as well as how nodes make routing decisions in TARF.

Structure and Exchange of Routing Information: A broadcast message from the base station fits into a fixed number of packets; in our implementation, it fits into one byte. Such a message consists of a few pairs of <the node id of a source node, an undelivered sequence interval [a, b] with a significant length>. To reduce overhead, only a few such pairs are selected to be broadcast. The undelivered sequence interval [a, b] is explained as follows: the base station searches the source sequence numbers received in the past few periods, identifies which source sequence numbers for the source node with this id are missing, and chooses certain significant interval [a, b] of missing source sequence numbers as an undelivered sequence interval. For example, the base station may have all the source sequence numbers for the source node 2 as {109, 110, 111, 150,151} in the past two periods. Then [112, 149] is an undelivered sequence interval. Since the base station is usually connected to a powerful platform such as a desktop, a program can be developed on that powerful platform to assist in recording all the source sequence numbers and finding undelivered sequence intervals. The reason for searching over more than one period is to identify as many undelivered data packets as possible. To illustrate that, consider this example: suppose the source sequence numbers of delivered data packets from node 2 are {1, 2, 3} for the 1st period and {200, 201, 203} for the 2nd period; then simply searching over a single period would not discover the undelivered packets unless every node is required to send a fixed number of data packets over each period.

Accordingly, each node in the network stores a table of <the node id of a source node, a forwarded sequence interval [a, b] with a significant length> in the past few periods. The data packets with the source node and the sequence numbers falling in this forwarded sequence interval [a, b] have already been forwarded by this node. When the node receives a broadcast message with undelivered sequence intervals, its *TrustManager* will be able to identify which data packets forwarded by this node are not delivered to the base station. Considering the overhead to store such a table, old entries will be deleted once the table is full.

Once a fresh broadcast message from the base station is received, a node immediately invalidates all the existing energy cost entries: it is ready to receive a new energy report from its neighbors and choose its new next-hop node afterwards. Also, it is going to select a node either after a timeout is reached or after it has received an energy cost report from some highly trusted candidates with acceptable energy cost. A node immediately broadcasts its energy cost to its neighbors only after it has selected a new next-hop node. That energy cost is computed by its *EnergyWatcher* (see Section 3.3). A natural question is which node starts reporting its energy cost first. For that, note that when the base station is sending a broadcast message, a side effect is that its neighbors receiving that message will also regard this as an energy report: the base station needs 0 amount of energy to reach itself. As long as the original base station is faithful, it will be viewed as a trustworthy candidate by *TrustManager* on the neighbors of the base station. Therefore, those neighbors will be the first nodes to decide their next-hop node, which is the base station; they will start reporting their energy cost once that decision is made.

Route Selection: Now, we introduce how TARF decides routes in a WSN. Each node N relies on its neighborhood table to select an optimal route, considering both energy consumption and reliability. TARF makes good efforts in excluding those nodes that misdirect traffic by exploiting the replay of routing information.

For a node N to select a route for delivering data to the base station, N will select an optimal next-hop node from its neighbors based on trust level and energy cost and forward the data to the chosen next-hop node immediately. The neighbors with trust levels below a certain threshold will be excluded from being considered as candidates. Among the remaining known neighbors, N will select as its next-hop node a neighbor b with the minimal value of $\frac{E_{Nb}}{T_{Nb}}$, with E_{Nb} and T_{Nb} being b's energy cost and trust level value in the neighborhood table respectively (see Section 3.3, 3.4). Basically, E_{Nb} reflects the energy cost of delivering a packet to the base station from N assuming that all the nodes in the route are honest; $\frac{1}{T_{Nb}}$ approximately reflects the number of the needed attempts to send a packet from N to the base station via multiple hops before such an attempt succeeds, considering the trust level of b. Thus, comparing the values of $\frac{E_{Nb}}{T_{Nb}}$ among N's neighbors identifies a candidate with a minimal combined cost of energy and trustworthiness.

The remaining delivery task is fully delegated to that selected next-hop neighbor, and N is totally unaware of what routing decision its chosen neighbor is going to make. Next, the chosen node will repeat what N has done, i.e., delegating the left routing task to its own chosen next-hop neighbor. In this way, instead of finding out a complete path to the base station, each node is only responsible for choosing its next-hop node,

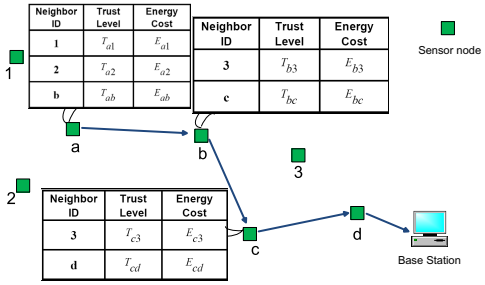


Fig. 3. Routing illustration

thus saving considerable cost in computation and routing information exchange. As an example shown in Figure 3, node a is trying to forward a packet to the base station. After comparing both the trust level and energy cost among its neighbors 1, 2 and b, a decides that b is the most promising next-hop node for data delivery and forwards the data packet to b immediately. b is free to make its own decision for routing the packet to the base station. b decides that its neighbor c is a better candidate than its neighbor 3. After that, the task is delegated to c, and c continues to delegate the job to d. Finally, d delivers the packet to the base station. Observe that in an ideal misbehavior-free environment, all nodes are absolutely faithful, and each node will choose a neighbor through which the routing path is optimized in terms of energy; thus, an energy-driven route is achieved. If we further assume that the one-hop transmission power of a unit-sized packet is the same for each node, the selected route will be the classical shortest path.

3.3 EnergyWatcher

Here we describe how a node N's *EnergyWatcher* computes the energy cost E_{Nb} for its neighbor b in N's neighborhood table and how N decides its own energy cost E_N. Before going further, we will clarify some notations. E_{Nb} mentioned is the average energy cost of successfully delivering a unit-sized data packet from N to the base station, with b as N's next-hop node being responsible for the remaining route. Here, one-hop re-transmission may occur until the acknowledgement is received or the number of re-transmissions reaches a certain threshold. The cost caused by one-hop re-transmissions should be included when computing E_{Nb}. Suppose N decides that A should be its next-hop node after comparing energy cost and trust level. Then N's energy cost is $E_N = E_{NA}$. Denote $E_{N\to b}$ as the average energy cost of successfully delivering a data packet from N to its neighbor b with one hop. Note that the re-transmission cost needs to be considered. With the above notations, it is straightforward to establish the following relation:

$$E_{Nb} = E_{N\to b} + E_b$$

Since each known neighbor b of N is supposed to broadcast its own energy cost E_b to N, to compute E_{Nb}, N still needs to know the value $E_{N\to b}$, i.e., the average energy cost of successfully delivering a data packet from N to its neighbor b with one hop. For that,

assuming that the endings (being acknowledged or not) of one-hop transmissions from N to b are independent with the same probability p_{succ} of being acknowledged, we first compute the average number of one-hop sendings needed before the acknowledgement is received as follows:

$$\sum_{i=1}^{\infty} i \cdot p_{succ} \cdot (1 - p_{succ})^{i-1} = \frac{1}{p_{succ}}$$

Denote E_{unit} as the energy cost for node N to send a unit-sized data packet once regardless of whether it is received or not. Then we have

$$E_{Nb} = \frac{E_{unit}}{p_{succ}} + E_b$$

The remaining job for computing E_{Nb} is to get the probability p_{succ} that a one-hop transmission is acknowledged. Considering the variable wireless connection among wireless sensor nodes, we do not use the simplistic averaging method to compute p_{succ}. Instead, after each transmission from N to b, N's *EnergyWatcher* will update p_{succ} based on whether that transmission is acknowledged or not with a weighted averaging technique. We use a binary variable Ack to record the result of current transmission: 1 if an acknowledgement is received; otherwise, 0. Given Ack and the last probability value of an acknowledged transmission p_{old_succ}, TARF uses a weighted average of Ack and p_{old_succ} as the new probability value p_{new_succ}:

$$p_{new\_succ} = (1 - w) \times p_{old\_succ} + w \times Ack, w \in (0, 1),$$

where w can be chosen by specific protocols.

3.4 TrustManager

A node N's *TrustManager* decides the trust level of each neighbor based on the following events: discovery of network loops, and broadcast from the base station about undelivered data packets. For each neighbor b of N, T_{Nb} denotes the trust level of b in N's neighborhood table. At the beginning, each neighbor is given a neutral trust level 0.5. After any of those events occurs, the relevant neighbors' trust levels are updated.

To detect loops, the *TrustManager* on N reuses the table of <the node id of a source node, a forwarded sequence interval [a, b] with a significant length> (see Section 3.2) in the past few periods. If N finds that a received data packet is already in that record table, not only will the packet be discarded, but the TrustManager on N also degrades its next-hop node's trust level. If that next-hop node is b, then T_{old_Nb} is the latest trust level value of b. We use a binary variable $Loop$ to record the result of loop discovery: 1 if a loop is received; 0 otherwise. After the degradation, as in the update of energy cost, the new trust level of b is

$$T_{new\_Nb} = (1 - w) \times T_{old\_Nb} + w \times Loop, w \in (0, 1),$$

where w can be chosen by specific applications.

Once a loop has been detected by N for a few times so that the trust level of the next-hop node is too low, N will change its next-hop selection; thus, that loop is broken. Though N can not tell which node should be held responsible for the occurrence of a loop, degrading its next-hop node's trust level gradually leads to the breaking of the loop.

On the other hand, to detect the traffic misdirection by nodes exploiting the replay of routing information, *TrustManager* on N compares N's stored table of <node id of a source node, forwarded sequence interval [a, b] with a significant length> recorded in the past few periods with the broadcast messages from the base station about undelivered data. It computes the ratio of the number of successfully delivered packets which are forwarded by this node to the number of those forwarded data packets, denoted as $DeliveryRatio$. Then N's *TrustManager* updates its next-hop node b's trust level as follows:

$$T_{new\_Nb} = (1 - w) \times T_{old\_Nb} + w \times DeliveryRatio, w \in (0, 1),$$

Now, suppose an adversary M forges the identity of the base station by replaying all the routing packets from the base station. At first, it is able to deceive its neighbors into believing that M is a base station; as a result, M may attract a large amount of data packets, which never reach the base station. However, after the base station broadcasts the information about those undelivered packets, M's neighbors will downgrade M's trust level values in their neighborhood table. Note that M is only capable of replaying but is not capable of manipulating or generating authenticated broadcast messages, and that M usually cannot prevent other nodes from receiving a broadcast message from the base station. As time elapses, M's neighbors will start realizing that M is not trustworthy and will look for other next-hop candidates that are more reliable. Similarly, if M forges the identity of another valid appealing node, M's neighbors will gradually realize that M is not reliable.

4 Implementation and Empirical Evaluation

We have implemented a protocol based on TARF in TinyOS 1.x, which currently runs on mica2 motes. Both the authentication and encryption of packets reuse the implementation of TinySec [7]: TinySec uses a CBC mode encryption scheme with Skipjack as the block cipher and an authentication scheme based on a four-byte message authentication code (MAC) computed by the CBC-MAC construction procedure. The MAC field is computed over the whole message including all the headers; it also serves as the CRC field of the packet. Data encryption can be disabled. In a routing packet, the next-hop id is replaced by a neighborhood broadcast address or a network broadcast address to indicate that it is a neighborhood or whole network broadcast. The acknowledgement of data packets is enabled. Considering the fact that floating-point computation is not supported by sensor hardware, the implementation uses an integer in $[0, 100]$ to represent trust level; the update of energy cost and trust level values is also implemented using integer arithmetics.

This implemented TARF protocol requires moderate program storage and memory usage. For comparison, we list the ROM size and RAM size requirement for this protocol and two other protocols on mica nodes in Table 1. The two other protocols are

Table 1. Size of protocol components implemented

Protocol	Authentication&Encryption	ROM (bytes)	RAM (bytes)
TARF	TinySec	20912	1464
Route	TinySec	20696	1048
MintRoute	TinySec	22554	1990

named Route and MintRoute according to their directory name under TinyOS 1.x. Both Route and MintRoute were the "standard" routing protocols in TinyOS 1.x and make route decisions based on both link quality estimation and number of hops. Neither of these original protocols provides encryption or authentication; to compare on a fair basis, we also enabled the encryption and authentication mode of TinySec for Route and MintRoute. TinySec occupies 728 bytes of RAM and 7146 bytes of ROM [7]. Similarly to Route and MintRoute, this TARF protocol adopts energy-efficient routes in a misbehavior-free environment. However, with a comparable size, it also supports the circumvention of adversaries exploiting the replay of routing information, which is not provided by Route or MintRoute. Further, our experience shows that it is easy to incorporate this TARF protocol into most applications. As an example, we re-implemented the Surge application in the TinyOS 1.x directory with this TARF protocol. The program has a size comparable to that of the Surge implemented using Route or MintRoute.

To evaluate how effective TARF is against deception through replaying routing information in the real world, we uploaded programs onto Motelab [13] at Harvard University. As a public test bed of wireless sensor networks, at the time of our experiments, 184 TMote Sky sensor motes were deployed at 3 floors. These nodes are distributed among many rooms of the building, with an approximate indoor transmission of 100 meters. Approximately 14 nodes were removed, and nearly 50 nodes were disabled. Motelab switched its serial forwarder protocol from TinyOS 1.x to TinyOS 2.x and was equipped with TMote only Tmote Sky motes. Due to the unavailability of TinySec on TMote SKy nodes, we did not include authentication or encryption from TinySec in the uploaded programs. Further, considering the availability of routing protocols on TinyOS 2.x, we compared our TinyOS 2.x version of TARF with the collection tree routing protocol (CTP), which mainly employs link quality estimation in choosing next-hop nodes. Both protocols were integrated into a data collection application - MultihopOscilloscope, which is named after its directory name in TinyOS 2.x. We configured the MultihopOscilloscope to send out 5 samples in a single data packet every 5 seconds. The routing update occurred every 50 seconds. Because of the limited quota assigned by Motelab, our programs lasted maximally 30 minutes. Among all the nodes, one was chosen to be the base station. Another node was programmed to be a fake base station: it broadcast as if it were a base station but never delivered the received data to the real base station. The many experiments we executed indicate that our TARF protocol achieves at least 30% higher *throughput* than CPT when there is an "attractive" fake base station. Some fake base stations are not able to misdirect much traffic because they have a poor wireless connection with their neighbors and do not look "appealing".

In one experiment (Figure 4(a)), all nodes on the three floors were supposed to deliver data to node 9 (the base station); node 15 (fake base station) replayed all the routing packets from the base station. By counting the data packets received at the real base station, TARF had approximately a 60% higher *throughput* than CTP. In another experiment (Figure 4(b)), only the nodes on the first floor (56 nodes totally) sent data to node 9 (the base station), and node 27 (fake base station) replayed the routing packets from the base station. As a result, TARF had approximately a 40% higher *throughput* than CTP.

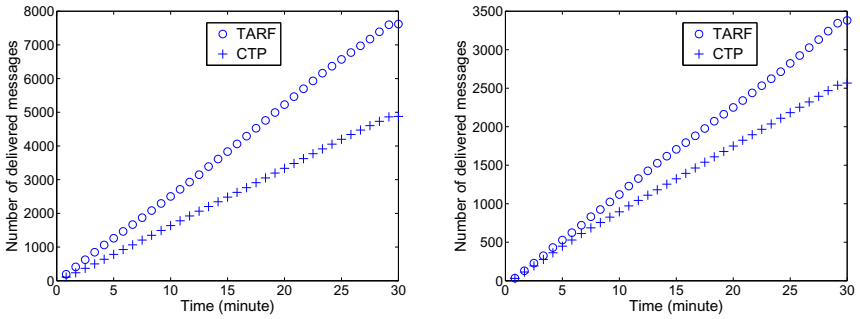


Fig. 4. With a fake base at Motelab, (a) TARF had approximately a 60% higher *throughput* than CTP among 3 floors; (b) TARF had approximately a 40% higher *throughput* than CTP at a single floor.

We also recorded the number of redundant data packets received by the base station. It turns out that both TARF and CTP had redundancy ratios at no more than 2%. Though both CTP and TARF suppress redundant packets, a packet might be received more than once by the base station because an acknowledgment is lost when the route changes.

5 Simulation and Evaluation

To further evaluate the efficacy of TARF in terms of energy efficiency and *throughput*, we have developed a reconfigurable emulator of wireless sensor networks on a two-dimensional plane with Matlab [12]. To effectively simulate a WSN, this emulator uses the object-oriented technique to construct two classes of objects: WSNMANAGER and NODE, to represent the whole network and a sensor node. The interaction between nodes are emulated through event passing. The routing function for a node can be rewritten to adopt different routing protocols; different maps can also be ported into this simulator. To simulate the unreliable wireless transmission, the outcome of one-hop packet transmission is decided by the following model: suppose a node A is wirelessly transmitting a packet to node B, the probability for B to successfully receive such a packet is assumed to be

$$1 - (min(dist, MAX\_DIST)/MAX\_DIST)^8,$$

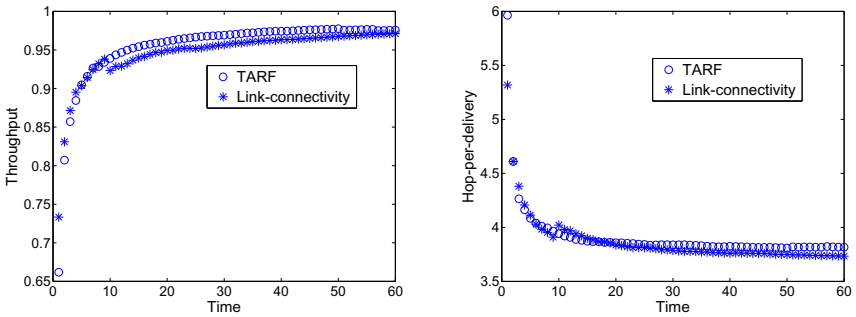
where $dist$ is the distance from A to B, and MAX_DIST is the maximal transmission range. In our experiment, MAX_DIST is defined as 100m, and 35 nodes are randomly distributed within a 300*300 rectangular area. All the nodes have the same power level and the same maximal transmission range of 100m. A base station is placed at the origin [0, 0]. We simulate the sensor network in 60 consecutive periods; each node samples data 6 times in each period.

The performance of TARF is compared to that proposed in [17] by Alec Woo, Terence Tong and David Culler. In that project, link connectivity is used as a cost metric for routing, which is found to be more cost-effective than the well-known shortest path protocol. We will simply refer to the latter protocol as link-connectivity. In our simulation experiments, we compare TARF with a simulated version of link-connectivity. As we will see from the experiment results, with the existence of misbehaviors, the *throughput* in TARF is often much higher than that in link-connectivity; the *hop-per-delivery* in TARF is generally at least comparable to that in the link connectivity protocol.

We compare TARF and link-connectivity under the following scenarios: (1) no nodes misbehaves intentionally; (2) certain nodes forge the identity of the based station by replaying broadcast messages; (3) a set of nodes colludes to form a forwarding loop; and (4) a set of nodes drops received data packets.

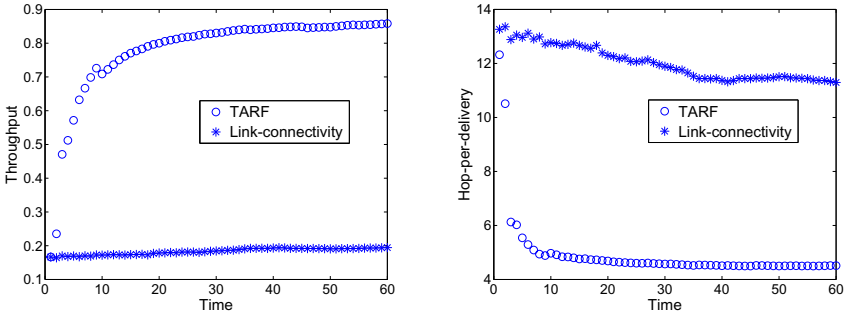
Under scenario (1) without misbehaving nodes, the two protocols have comparable performance in terms of *throughput* and *hop-per-delivery*. Figure 5 shows such an example. Under a misbehavior-free environment, according to the TARF protocol, a node may still perceive its neighbors as having different trust level, due to the fact that the node can not well distinguish between malicious behavior and failed delivery due to environmental effects. However, such mis-perception of trust does not compromise the performance of TARF.

Under scenario (2), certain malicious nodes become fake base stations through replaying messages originated from the base station. With the link connectivity protocol, a significant portion of traffic is attracted to the fake base. However, with TARF, most packets are able to select a route circumventing those fake bases. When there are forged base stations, TARF tends to show much better *throughput* than the link



(a) (b)

Fig. 5. Under misbehavior-free environment, TARF and link-connectivity have comparable performance in (a) *throughput*, and (b) *hop-per-delivery*.



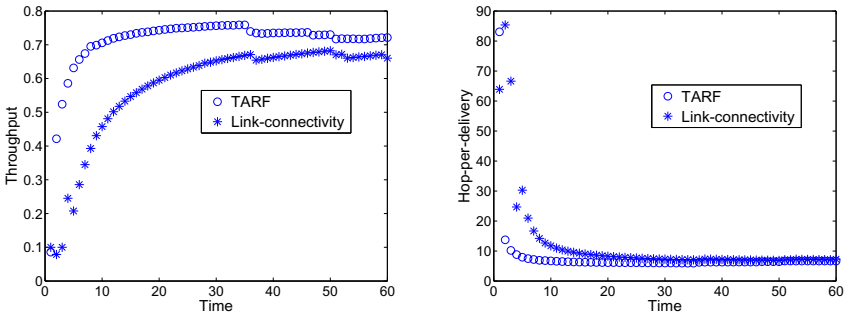
(a) (b)

Fig. 6. With a fake base, (a) TARF has 5 times the *throughput* in link-connectivity; (b) TARF has less than 50% *hop-per-delivery* in link-connectivity.

connectivity protocol, and the *hop-per-delivery* in TARF is much less than that in the link-connectivity protocol. In one of our experiments with a fake base station, as indicated in Figure 6, TARF reaches roughly 5 times the *throughput* in the link-connectivity protocol, while the *hop-per-delivery* in TARF is less than 50% that in link-connectivity.

Under scenario (3), a loop of colluding nodes intercepts many packets. The *throughput* in TARF is generally higher than that in link-connectivity; the *hop-per-delivery* in the two protocols gradually become comparable. In one experiment, as shown in Figure 7, 5 out of 35 nodes are selected to form a network loop. Any data forwarded to one of these 6 nodes would not be able to arrive at the base station. As in Figure 7, the *throughput* in TARF is around 70% higher than that in the link connectivity protocol; their *hop-per-delivery* gradually becomes comparable.

Under scenario (4), a set of nodes drops any received data packets. In our experiment, 6 nodes drop data forwarded to them. As indicated by Figure 8, the *throughput* in TARF



(a) (b)

Fig. 7. With a loop consisting of 14% nodes, (a) TARF has a higher *throughput* than link-connectivity; (b) gradually, TARF and link-connectivity have comparable *hop-per-delivery*.

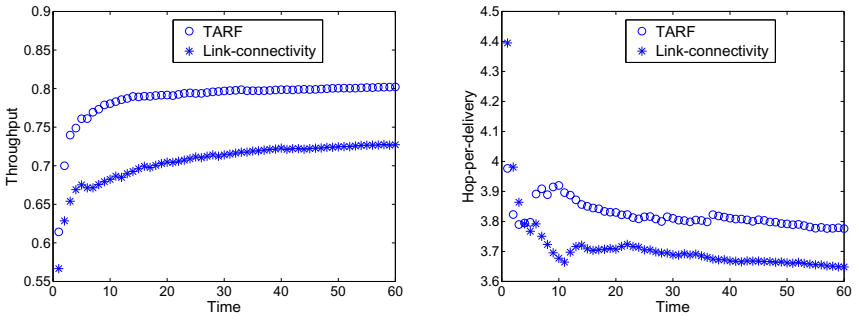


Fig. 8. With 6 nodes dropping data, (a) TARF has a 14% higher *throughput* than link-connectivity; (b) TARF has a 5% higher *hop-per-delivery* than link-connectivity.

is at least 14% greater than that in link-connectivity; the *hop-per-delivery* in TARF is around 5% higher than that in link-connectivity.

6 Conclusions

We propose TARF, a trust-aware routing framework for WSNs, to secure multi-hop routing in WSNs against intruders exploiting the replay of routing information. With the idea of trust management, TARF enables a node to keep track of the trustworthiness of its neighbors and thus to select a reliable route. Not only does TARF circumvent those malicious nodes misusing other nodes' identities to misdirect network traffic, it also accomplishes efficient energy usage. Our implementation and simulation results indicate that (1) the efficiency of energy usage in TARF is generally at least comparable to that in existing protocols; (2) with the existence of traffic misdirection through "identity theft", TARF generally achieves a significantly higher *throughput* than other existing protocols; and (3) TARF is scalable and adaptable to typical medium-scale testbed environments and simulated conditions. Our future work is to further evaluate TARF with large-scale WSNs deployed in wild environments and to study how to choose parameters involved for specific applications. We believe that the idea of TARF can also be applied to general ad hoc networks and peer-to-peer networks to fight against similar attacks.

References

1. Al-Karaki, J., Kamal, A.: Routing techniques in wireless sensor networks: a survey. IEEE Wireless Communications 11(6), 6–28 (2004)
2. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of 1996 IEEE Symposium on Security and Privacy, pp. 164–173 (1996)
3. Boukerche, A., El-Khatib, K., Xu, L., Korba, L.: A novel solution for achieving anonymity in wireless ad hoc networks. In: Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, pp. 30–38 (2004)

4. Ganeriwal, S., Balzano, L., Srivastava, M.: Reputation-based framework for high integrity sensor networks. ACM Trans. Sen. Netw. (2008)
5. He, Q., Wu, D., Khosla, P.: Sori: A secure and objective reputation-based incentive scheme for ad hoc networks. In: Proceedings of IEEE Wireless Communications and Networking Conference, pp. 825–830 (2004)
6. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web, pp. 640–651 (2003)
7. Karlof, C., Sastry, N., Wagner, D.: Tinysec: A link layer security architecture for wireless sensor networks. In: Proc. of ACM SenSys 2004 (November 2004)
8. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. In: First IEEE International Workshop on Sensor Network Protocols and Applications (2003)
9. Liang, Z., Shi, W.: Pet: A personalized trust model with reputation and risk evaluation for p2p resource sharing. In: HICSS 2005: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS 2005) - Track 7. IEEE Computer Society, Los Alamitos (2005)
10. Liu, A., Ning, P.: Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In: IPSN 2008: Proceedings of the 7th international conference on Information processing in sensor networks, pp. 245–256. IEEE Computer Society, Los Alamitos (2008)
11. Liu, Z., Joy, A., Thompson, R.: A dynamic trust model for mobile ad hoc networks. In: Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, pp. 80–85 (2004)
12. Matlab, `http://www.mathworks.com`
13. Motelab, `http://motelab.eecs.harvard.edu`
14. Perrig, A., Szewczyk, R., Wen, W., Culler, D., Tygar, J.: SPINS: Security protocols for sensor networks. Wireless Networks Journal (WINET) 8(5), 521–534 (2002)
15. Wang, Y., Vassileva, J.: Trust and reputation model in peer-to-peer networks. In: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, p. 150 (2003)
16. Watro, R., Kong, D., Cuti, S., Gardiner, C., Lynn, C., Kruus, P.: Tinypk: securing sensor networks with public key technology. In: SASN 2004: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, pp. 59–64. ACM, New York (2004)
17. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the First ACM SenSys 2003 (November 2003)
18. Wood, A., Stankovic, J.: Denial of service in sensor networks. Computer 35(10), 54–62 (2002)
19. Zhan, G., Shi, W., Deng, J.: Poster abstract: Sensortrust - a resilient trust model for wsns. In: SenSys 2009: Proceedings of the 7th International Conference on Embedded Networked Sensor Systems (2009)