Design and Implementation of TARF: A Trust-Aware Routing Framework for WSNs

Guoxing Zhan, Weisong Shi, *Senior Member, IEEE*, and Julia Deng

Abstract—The multi-hop routing in wireless sensor networks (WSNs) offers little protection against identity deception through replaying routing information. An adversary can exploit this defect to launch various harmful or even devastating attacks against the routing protocols, including *sinkhole* attacks, wormhole attacks and *Sybil* attacks. The situation is further aggravated by mobile and harsh network conditions. Traditional cryptographic techniques or efforts at developing trust-aware routing protocols do not effectively address this severe problem. To secure the WSNs against adversaries misdirecting the multi-hop routing, we have designed and implemented TARF, a robust trust-aware routing framework for dynamic WSNs. Without tight time synchronization or known geographic information, TARF provides trustworthy and energy-efficient route. Most importantly, TARF proves effective against those harmful attacks developed out of identity deception; the resilience of TARF is verified through extensive evaluation with both simulation and empirical experiments on large-scale WSNs under various scenarios including mobile and RF-shielding network conditions. Further, we have implemented a low-overhead TARF module in TinyOS; as demonstrated, this implementation can be incorporated into existing routing protocols with the least effort. Based on TARF, we also demonstrated a proof-of-concept mobile target detection application that functions well against an anti-detection mechanism.

1 Introduction

Wireless sensor networks (WSNs) [2] are ideal candidates for applications to report detected events of interest, such as military surveillance and forest fire monitoring. A WSN comprises battery-powered senor nodes with extremely limited processing capabilities. 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. An 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 [3]. This paper focuses on the kind of attacks in which adversaries misdirect network traffic by identity deception through replaying routing information. Based on identity deception, the adversary is capable of launching harmful and hard-to-detect attacks against routing, such as *selective forwarding*, *wormhole* attacks, *sinkhole* attacks and *Sybil* attacks [4].

As a harmful and easy-to-implement type of attack, a malicious node simply replays all the outgoing routing packets from a valid node to forge the latter node's identity; the malicious node then uses this forged identity to participate in the network routing, thus disrupting the network traffic. Those routing 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 the original valid node, which is known as a *wormhole* attack [5]. 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. For instance, 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 [4]. *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 [6]. Such a fake base station could lure more than half the traffic, creating a "black hole". This same technique can be employed to conduct another strong form of attack - *Sybil* attack [7]: through replaying the routing information of multiple legitimate nodes, an attacker may present multiple identities to the network. A valid node, if compromised, can also launch all these attacks.

The harm of such malicious attacks based on the technique of replaying routing information is further aggravated by the introduction of mobility into WSNs and the hostile network condition. Though mobility is introduced into WSNs for efficient data collection and

- *G. Zhan and W. Shi are with the Department of Computer Science, Wayne State University, Detroit, MI, 48202. E-mail: gxzhan@wayne.edu, weisong@wayne.edu.*
- *J. Deng is with Intelligent Automation Inc., Rockville, MD 20855. Email: hdeng@i-a-i.com.*

This work is in part supported by AFRL contract FA8650-10-C-1740 and NSF Career Award CCF-0643521. We also would like to thank the program manager Mr. John Woods for his great support as well as the anonymous reviewers for their constructive comments and suggestions.

various applications [8], [9], [10], [11], it greatly increases the chance of interaction between the honest nodes and the attackers. Additionally, a poor network connection causes much difficulty in distinguishing between an attacker and a honest node with transient failure. Without proper protection, WSNs with existing routing protocols can be completely devastated under certain circumstances. In an emergent sensing application through WSNs, saving the network from being devastated becomes crucial to the success of the application.

Unfortunately, most existing routing protocols for WSNs either assume the honesty of nodes and focus on energy efficiency [12], or attempt to exclude unauthorized participation by encrypting data and authenticating packets. Examples of these encryption and authentication schemes for WSNs include TinySec [13], Spins [14], TinyPK [15], and TinyECC [16]. Admittedly, it is important to consider efficient energy use for battery-powered sensor nodes and the robustness of routing under topological changes as well as common faults in a wild environment. However, it is also critical 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. The gossiping-based routing protocols offer certain protection against attackers by selecting random neighbors to forward packets [17], but at a price of considerable overhead in propagation time and energy use.

In addition to the cryptographic methods, trust and reputation management has been employed in generic ad hoc networks and WSNs to secure routing protocols. Basically, a system of trust and reputation management assigns each node a trust value according to its past performance in routing. Then such trust values are used to help decide a secure and efficient route. However, the proposed trust and reputation management systems for generic ad hoc networks target only relatively powerful hardware platforms such as laptops and smartphones [18], [19], [20], [21]. Those systems can not be applied to WSNs due to the excessive overhead for resource-constrained sensor nodes powered by batteries. As far as WSNs are concerned, secure routing solutions based on trust and reputation management rarely address the identity deception through replaying routing information [22], [23]. The countermeasures proposed so far strongly depends on either tight time synchronization or known geographic information while their effectiveness against attacks exploiting the replay of routing information has not been examined yet [4].

At this point, to protect WSNs from the harmful attacks exploiting the replay of routing information, we have designed and implemented a robust trust-aware routing framework, TARF, to secure routing solutions in wireless sensor networks. Based on the unique characteristics of resource-constrained WSNs, the design of TARF centers on *trustworthiness* and *energy efficiency*. Though TARF can be developed into a complete and independent routing protocol, the purpose is to allow existing routing protocols to incorporate our implementation of TARF with the least effort and thus producing a secure and efficient fully-functional protocol. Unlike other security measures, TARF requires neither tight time synchronization nor known geographic information. Most importantly, TARF proves resilient under various attacks exploiting the replay of routing information, which is not achieved by previous security protocols. Even under strong attacks such as *sinkhole* attacks, *wormhole* attacks as well as *Sybil* attacks, and hostile mobile network condition, TARF demonstrates steady improvement in network performance. The effectiveness of TARF is verified through extensive evaluation with simulation and empirical experiments on large-scale WSNs. Finally, we have implemented a ready-to-use TARF module with low overhead, which as demonstrated can be integrated into existing routing protocols with ease; the demonstration of a proof-of-concept mobile target detection program indicates the potential of TARF in WSN applications.

We start by stating the design considerations of TARF in Section 2. Then we elaborate the design of TARF in Section 3, including the routing procedure as well as the *EnergyWatcher* and *TrustManager* components. In Section 4, we present the simulation results of TARF against various attacks through replaying routing information in static, mobile and RF-shielding conditions. Section 5 further presents the implementation of TARF, empirical evaluation at a large sensor network and a resilient proof-of-concept mobile target detection application based on TARF. Finally, we discuss the related work in Section 6 and conclude this paper in Section 7.

2 Design Considerations

Before elaborating the detailed design of TARF, we would like to clarify a few design considerations first, including certain assumptions in Section 2.1 and the goals in Section 2.3.

2.1 Assumptions

We target secure routing for data collection tasks, which are one of the most fundamental functions of WSNs. In a data collection task, a sensor node sends its sampled data to a remote base station with the aid of other intermediate nodes, as shown in Figure 1. 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 assume that there is only one base station. An adversary may forge the identity of any legal node through replaying that node's outgoing routing packets and spoofing the acknowledgement packets, even remotely through a *wormhole*.

Additionally, to merely simplify the introduction of TARF, we assume no data aggregation is involved.

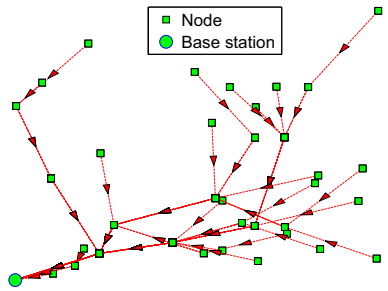


Fig. 1. Multi-hop routing for data collection of a WSN.

Nonetheless, our approach can still be applied to cluster-based WSNs with static clusters, where data are aggregated by clusters before being relayed [24]. Cluster-based WSNs allows for the great savings of energy and bandwidth through aggregating data from children nodes and performing routing and transmission for children nodes. In a cluster-based WSN, the 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 the cluster headers. Our framework can then be applied to this sub-network to achieve secure routing for cluster-based WSNs. TARF may run on cluster headers only and the cluster headers communicate with their children nodes directly since a static cluster has known relationship between a cluster header and its children nodes, though any link-level security features may be further employed.

Finally, we assume a data packet has 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 because that allows the base station to track whether a data packet is delivered. It would cause too much overhead to transmit all the one-hop information to the base station. Also, we assume the routing packet is sequenced.

2.2 Authentication Requirements

Though a specific application may determine whether data encryption is needed, TARF requires that the packets are properly authenticated, especially the broadcast packets from the base station. The broadcast from the base station is asymmetrically authenticated so as to guarantee that an adversary is not able to manipulate or forge a broadcast message from the base station at will. Importantly, with authenticated broadcast, even with the existence of attackers, TARF may use TrustManager (Section 3.4) and the received broadcast packets about delivery information (Section 3.2.1) to choose trustworthy path by circumventing compromised nodes. Without being able to physically capturing the base station, it is generally very difficult for the adversary to manipulate the base station broadcast packets which are asymmetrically authenticated. The asymmetric authentication of those broadcast packets from the base station is crucial to any successful secure routing protocol. It can be achieved through existing asymmetrically authenticated broadcast schemes that may require loose time synchronization. As an example, μTESLA [14] achieves asymmetric authenticated broadcast through a symmetric cryptographic algorithm and a loose delay schedule to disclose the keys from a key chain. Other examples of asymmetric authenticated broadcast schemes requiring either loose or no time synchronization are found in [25], [26].

Considering the great computation cost incurred by a strong asymmetric authentication scheme and the difficulty in key management, a regular packet other than a base station broadcast packet may only be moderately authenticated through existing symmetric schemes with a limited set of keys, such as the message authentication code provided by TinySec [13]. It is possible that an adversary physically captures a non-base legal node and reveals its key for the symmetric authentication [27]. With that key, the adversary can forge the identity of that non-base legal node and joins the network "legally". However, when the adversary uses its fake identity to falsely attract a great amount of traffic, after receiving broadcast packets about delivery information, other legal nodes that directly or indirectly forwards packets through it will start to select a more trustworthy path through TrustManager (Section 3.4).

2.3 Goals

TARF mainly guards a WSN against the attacks misdirecting the multi-hop routing, especially those based on identity theft through replaying the routing information. This paper does not address the denial-of-service (DoS) [3] attacks, where an attacker intends to damage the network by exhausting its resource. For instance, we do not address the DoS attack of congesting the network by replaying numerous packets or physically jamming the network. TARF aims to achieve the following desirable properties:

High Throughput *Throughput* is defined as the ratio of the number of all data packets delivered to the base station to the number of all sampled data packets. In our evaluation, *throughput* at a moment is computed over the period from the beginning time (0) until that particular moment. Note that single-hop re-transmission may happen, and that duplicate packets are considered as one packet as far as *throughput* is concerned. *Throughput* reflects how efficiently the network is collecting and delivering data. Here we regard high *throughput* as one of our most important goals.

Energy Efficiency Data transmission accounts for a major portion of the 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 that each delivery of a data packet takes, abbreviated as *hop-per-delivery*.

Scalability & Adaptability TARF should work well with WSNs of large magnitude under highly dynamic contexts. We will evaluate the scalability and adaptability of TARF through experiments with large-scale WSNs and under mobile and hash network conditions.

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 incorporating TARF.

3 Design of TARF

TARF secures the multi-hop routing in WSNs against intruders misdirecting the multi-hop routing by evaluating the trustworthiness of neighboring nodes. It identifies such intruders 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

For a TARF-enabled 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 considering both the trustworthiness and the energy efficiency. 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. 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. The technique of maintaining a neighborhood table of a moderate size is demonstrated by Woo, Tong and Culler [28]; TARF may employ the same technique.

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 data delivery and energy cost report messages from each node. Neither message needs acknowledgement. A broadcast message from the base station is flooded to the whole network. The freshness of a broadcast message is checked through 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. 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. A compromised node may falsely report an extremely low energy cost to lure its neighbors into selecting this compromised node as their next-hop node; however, these TARF-enabled neighbors eventually abandon that compromised next-hop node based on its low trustworthiness as tracked by *TrustManager*. *TrustManager* is responsible for tracking trust level values of neighbors based on network loop discovery and broadcast messages from the base station about data delivery. 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*.

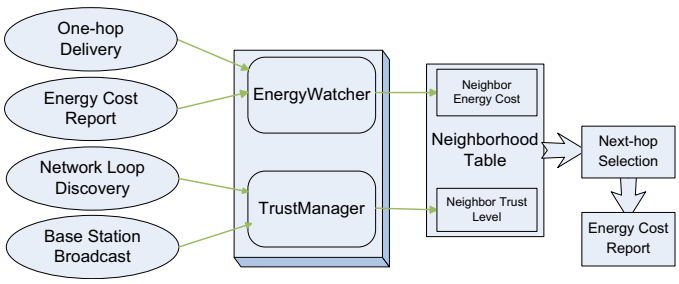


Fig. 2. Each node selects a next-hop node based on its neighborhood table, and broadcast its energy cost within its neighborhood. To maintain this neighborhood table, *EnergyWatcher* and *TrustManager* on the node keep track of related events (on the left) to record the energy cost and the trust level 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 a message about data delivery during last period to the whole network consisting of a few contiguous packets (one packet may not hold all the information). Each such packet has a field to indicate how many packets are remaining to complete the broadcast of the current message. The completion of the base station broadcast triggers the exchange of energy report 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. No tight 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 data delivery 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 message from the base station. If a node does not change its next-hop node selection until the next broadcast message 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 when its current next-hop node performs the task of receiving and delivering data poorly.

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

3.2.1 Structure and Exchange of Routing Information

A broadcast message from the base station fits into at most a fixed small number of packets. Such a message consists of some pairs of <node id of a source node, an undelivered sequence interval [a, b] with a significant length>, <node id of a source node, minimal sequence number received in last period, maximum sequence number received in last period>, as well as several node id intervals of those without any delivery record in last period. To reduce overhead to an acceptable amount, our implementation selects only a limited number of such pairs to broadcast (Section 5.1) and proved effective (Section 5.3, 5.4). Roughly, the effectiveness can be explained as follows: the fact that an attacker attracts a great deal of traffic from many nodes often gets revealed by at least several of those nodes being deceived with a high likelihood. The undelivered sequence interval [a, b] is explained as follows: the base station searches the source sequence numbers received in last period, 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 last period. Then [112, 149] is an undelivered sequence interval; [109, 151] is also recorded as the sequence boundary of delivered packets. 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.

Accordingly, each node in the network stores a table of <node id of a source node, a forwarded sequence interval [a, b] with a significant length> about last period. 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 about data delivery, 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.

3.2.2 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 its next-hop node through evaluating each neighbor b based on a trade-off between T_{Nb} and $\frac{E_{Nb}}{T_{Nb}}$, 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, $\frac{E_{Nb}}{T_{Nb}}$ combines the trustworthiness and energy cost. However, the metric $\frac{E_{Nb}}{T_{Nb}}$ suffers from the fact that an adversary may falsely reports extremely low energy cost to attract traffic and thus resulting in a low value of $\frac{E_{Nb}}{T_{Nb}}$ even with a low T_{Nb}. Therefore, TARF prefers nodes with significantly higher trust values; this preference of trustworthiness effectively protects the network from an adversary who forges the identity of an attractive node such as a base station. For deciding the next-hop node, a specific trade-off between T_{Nb} and $\frac{E_{Nb}}{T_{Nb}}$ is demonstrated in Figure 5 (see Section 5.2).

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.

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}, an intuitive way is to use a simply weighted average of Ack and p_{old_succ} as the value of p_{new_succ}. That is what is essentially adopted in the aging mechanism [29]. However, that method used against sleeper attacks still suffers periodic attacks [30]. To solve this problem, we update the p_{succ} value using two different weights as in our previous work [30], a relatively big $w_{degrade} \in (0, 1)$ and a relatively small $w_{upgrade} \in (0, 1)$ as follows:

$$p_{new\_succ} = \begin{cases} (1 - w_{degrade}) \times p_{old\_succ} + w_{degrade} \times Ack, \\ \text{if } Ack = 0. \\ (1 - w_{upgrade}) \times p_{old\_succ} + w_{upgrade} \times Ack, \\ \text{if } Ack = .1 \end{cases}$$

The two parameters $w_{degrade}$ and $w_{upgrade}$ allow flexible application requirements. $w_{degrade}$ and $w_{upgrade}$ represent the extent to which upgraded and degraded performance are rewarded and penalized, respectively. If any fault and compromise is very likely to be associated with a high risk, $w_{degrade}$ should be assigned a relatively high value to penalize fault and compromise relatively heavily; if a few positive transactions can't constitute evidence of good connectivity which requires many more positive transactions, then $w_{upgrade}$ should be assigned a relatively low value.

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 data delivery. 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.

Note that many existing routing protocols have their own mechanisms to detect routing loops and to react accordingly [31], [32], [28]. In that case, when integrating TARF into those protocols with anti-loop mechanisms, *TrustManager* may solely depend on the broadcast from the base station to decide the trust level; we adopted such a policy when implementing TARF later (see Section 5). If anti-loop mechanisms are both enforced in the TARF component and the routing protocol that integrates TARF, then the resulting hybrid protocol may overly react towards the discovery of loops. Though sophisticated loop-discovery methods exist in the currently developed protocols, they often rely on the comparison of specific routing cost to reject routes likely leading to loops [32]. To minimize the effort to integrate TARF and the existing protocol and to reduce the overhead, when an existing routing protocol does not provide any anti-loop mechanism, we adopt the following mechanism to detect routing loops. To detect loops, the *TrustManager* on N reuses the table of <node id of a source node, a forwarded sequence interval [a, b] with a significant length> (see Section 3.2) in last period. 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: 0 if a loop is received; 1 otherwise. As in the update of energy cost, the new trust level of b is

$$T_{new\_Nb} = \begin{cases} (1 - w_{degrade}) \times T_{old\_Nb} + w_{degrade} \times Loop, \\ \text{if } Loop = 0. \\ (1 - w_{upgrade}) \times T_{old\_Nb} + w_{upgrade} \times Loop, \\ \text{if } Loop = 1. \end{cases}$$

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 last period with the broadcast messages from the base station about data delivery. 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:

$$T_{new\_Nb} = \begin{cases} (1 - w_{degrade}) \times T_{old\_Nb} \\ +w_{degrade} \times DeliveryRatio, \\ \text{if } DeliveryRatio < T_{old\_Nb}. \\ \\ (1 - w_{upgrade}) \times T_{old\_Nb} \\ +w_{upgrade} \times DeliveryRatio, \\ \text{if } DeliveryRatio >= T_{old\_Nb}. \end{cases}$$

3.5 Analysis on EnergyWatcher and TrustManager

Now that a node N relies on its *EnergyWatcher* and *TrustManager* to select an optimal neighbor as its next-hop node, we would like to clarify a few important points on the design of *EnergyWatcher* and *TrustManager*.

First, as described in Section 3.1, the energy cost report is the only information that a node is to passively receive and take as "fact". It appears that such acceptance of energy cost report could be a pitfall when an attacker or a compromised node forges false report of its energy cost. Note that the main interest of an attacker is to prevent data delivery rather than to trick a data packet into a less efficient route, considering the effort it takes to launch an attack. As far as an attack aiming at preventing data delivery is concerned, TARF well mitigates the effect of this pitfall through the operation of *TrustManager*. Note that the *TrustManager* on one node does not take any recommendation from the *TrustManager* on another node. If an attacker forges false energy report to form a false route, such intention will be defeated by *TrustManager*: when the *TrustManager* on one node finds out the many delivery failures from the broadcast messages of the base station, it degrades the trust level of its current next-hop node; when that trust level goes below certain threshold, it causes the node to switch to a more promising next-hop node.

Second, *TrustManager* identities the low trustworthiness of various attackers misdirecting the multi-hop routing, especially those exploiting the replay of routing information. It is noteworthy that *TrustManager* does not distinguish whether an error or an attack occurs to the next-hop node or other succeeding nodes in the route. It seems unfair that *TrustManager* downgrades the trust level of an honest next-hop node while the attack occurs somewhere after that next-hop node in the route. Contrary to that belief, *TrustManager* significantly improves data delivery ratio in the existence of attack attempts of preventing data delivery. First of all, it is often difficult to identify an attacker who participates in the network using an id "stolen" from another legal node. For example, it is extremely difficult to detect a few attackers colluding to launch a combined *wormhole* and *sinkhole* attack [4]. Additionally, despite the certain inevitable unfairness involved, *TrustManager* encourages a node to choose another route when its current route frequently fails to

deliver data to the base station. Though only those legal neighboring nodes of an attacker might have correctly identified the adversary, our evaluation results indicate that the strategy of switching to a new route without identifying the attacker actually significantly improves the network performance, even with the existence of *wormhole* and *sinkhole* attacks. Fig 3 gives an example to illustrate this point. In this example, node A, B, C and D are all honest nodes and not compromised. Node A has node B as its current next-hop node while node B has an attacker node as its next-hop node. The attacker drops every packet received and thus any data packet passing node A will not arrive at the base station. After a while, node A discovers that the data packets it forwarded did not get delivered. The *TrustManager* on node A starts to degrade the trust level of its current next-hop node B although node B is absolutely honest. Once that trust level becomes too low, node A decides to select node C as its new next-hop node. In this way node A identifies a better and successful route (A - C - D - base). In spite of the sacrifice of node B's trust level, the network performs better. Further, concerning the stability of routing path,

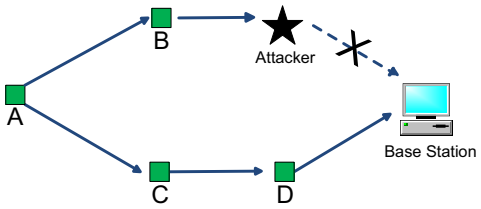


Fig. 3. An example to illustrate how *TrustManager* works.

once a valid node identifies a trustworthy honest neighbor as its next-hop node, it tends to keep that next-hop selection without considering other seemingly attractive nodes such as a fake base station. That tendency is caused by both the preference to maintain stable routes and the preference to highly trustable nodes.

Finally, we would like to stress that TARF is designed to guard a WSN against the attacks misdirecting the multi-hop routing, especially those based on identity theft through replaying the routing information. Other types of attacks such as the denial-of-service (DoS) [3] attacks are out of the discussion of this paper. For instance, we do not address the attacks of injecting into the network a number of data packets containing false sensing data but authenticated (possibly through hacking). That type of attacks aim to exhaust the network resource instead of misdirecting the routing. However, if the attacker intends to periodically inject a few routing packets to cause wrong route, such attacks can still be defended by TARF through *TrustManager*.

4 Simulation

We have developed a reconfigurable emulator of wireless sensor networks on a two-dimensional plane with Matlab to test TARF. We have conducted extensive simulation experiments; however, due to the page limit, interested readers may refer to our technical report [33] and the conference version of paper [1] for detailed simulation settings and experimental results. In our experiments, initially, 35 nodes are randomly distributed within a 300*300 rectangular area, with unreliable wireless transmission. All the nodes have the same power level and the same maximal transmission range of 100m. Each node samples 6 times in every period; the timing gap between every two consecutive samplings of the same node is equivalent. We simulate the sensor network in 1440 consecutive periods.

Regarding the network topology, we set up three types of network topologies. The first type is the static-location case under which all nodes stand still. The second type is a customized group-motion-with-noise case based on Reference Point Group Mobility (RPGM) model that mimics the behavior of a set of nodes moving in one or more groups [34], [35]. The last type of dynamic network incorporated in the experiments is the addition of scattered RF-shielded areas to the aforementioned group-motion-with-noise case.

The performance of TARF is compared to that of a link connectivity-based routing protocol adapted from what is proposed by Alec Woo, Terence Tong and David Culler [28]. We denote the link connectivity-based routing protocol as *Link-connectivity*. With the *Link-connectivity* protocol, each node selects its next-hop node among its neighborhood table according to an link estimator based on exponentially weighted moving average (EWMA). The simulation results show, in the presence of misbehaviors, the *throughput* in TARF is often much higher than that in *Link-connectivity*; the *hop-per-delivery* in the *Link-connectivity* protocol is generally at least comparable to that in TARF.

Under a misbehavior-free environment, the simulation results show that TARF and *Link-connectivity* have comparable performance when there is no adversary. Both protocols are also evaluated under three common types of attacks: (1) a certain node forges the identity of the based station by replaying broadcast messages, also known as the *sinkhole* attack; (2) a set of nodes colludes to form a forwarding loop; and (3) a set of nodes drops received data packets. These experiments were conducted in the static case, the group-motion-with-noise case, and the addition of RF-shielded areas to the group-motion-with-noise case separately. Generally, under these common attacks, TARF produces a substantial improvement over *Link-connectivity* in terms of data collection and energy efficiency. Further, we have evaluated TARF under more severe attacks: multiple moving fake bases and multiple *Sybil* attackers. As before, the experiments are conducted under all the three types of network topology. Under these two types of most severe attacks which almost devastates the *Link-connectivity* protocol, TARF succeeds in achieving a steady improvement over the *Link-connectivity* protocol. Finally, we have conducted

certain experiments to explore the choice of the period length and the trust updating scheme. Our experiments reveal that a shorter period or a faster trust updating scheme may not necessarily benefit TARF.

5 Implementation and Empirical Evaluation

In order to evaluate TARF in a real-world setting, we implemented the *TrustManager* component on TinyOS 2.x, which can be integrated into the existing routing protocols for WSNs with the least effort. Originally, we had implemented TARF as a self-contained routing protocol [1] on TinyOS 1.x before this second implementation. However, we decided to re-design the implementation considering the following factors. First, the first implementation only supports TinyOS 1.x, which was replaced by TinyOS 2.x; the porting procedure from TinyOS 1.x to TinyOS 2.x tends to frustrate the developers. Second, rather than developing a self-contained routing protocol, the second implementation only provides a *TrustManager* component that can be easily incorporated into the existing protocols for routing decisions. The detection of routing loops and the corresponding reaction are excluded from the implementation of *TrustManager* since many existing protocols, such as Collection Tree Protocol [32] and the link connectivity-based protocol [28], already provide that feature. As we worked on the first implementation, we noted that the existing protocols provide many nice features, such as the analysis of link quality, the loop detection and the routing decision mainly considering the communication cost. Instead of providing those features, our implementation focuses on the trust evaluation based on the base broadcast of the data delivery, and such trust information can be easily reused by other protocols. Finally, instead of using TinySec [13] exclusively for encryption and authentication as in the first implementation on TinyOS 1.x, this re-implementation let the developers decide which encryption or authentication techniques to employ; the encryption and authentication techniques of TARF may be different than that of the existing protocol.

5.1 TrustManager Implementation Details

The *TrustManager* component in TARF is wrapped into an independent TinyOS configuration named `TrustManagerC`. `TrustManagerC` uses a dedicated logic channel for communication and runs as a periodic service with a configurable period, thus not interfering with the application code. Though it is possible to implement TARF with a period always synchronized with the routing protocol's period, that would cause much intrusion into the source code of the routing protocol. The current `TrustManagerC` uses a period of 30 seconds; for specific applications, by modifying a certain header file, the period length may be re-configured to reflect the sensing frequency, the energy efficiency and trustworthiness requirement. `TrustManagerC` provides two interfaces (see Figure 4), `TrustControl` and `Record`, which are implemented in other modules. The `TrustControl` interface provides the commands to enable and disable the trust evaluation, while the `Record` interface provides the commands for a root, i.e., a base station, to add delivered message record, for a non-root node to add forwarded message record, and for a node to retrieve the trust level of any neighboring node. The implementation on a root node differs from that on a non-root node: a root node stores the information of messages received (delivered) during the current period into a record table and broadcast delivery failure record; a non-root node stores the information of forwarded messages during the current period also in a record table and compute the trust of its neighbors based on that and the broadcast information. Noting that much implementation overhead for a root can always be transferred to a more powerful device connected to the root, it is reasonable to assume that the root would have great capability of processing and storage.

```
configuration TrustManagerC {
  provides {
    interface TrustControl;
    interface Record;
  }
  implementation {
    .....
  }
}
```

```
interface TrustControl
{
  //enable trust evaluation
  command error_t start();
  //disable trust evaluation
  command error_t stop();
}
```

```
interface Record
{
  //for a root to add delivered record <source node id, source sequence number>
  command void addDelivered(am_addr_t src, uint8_t seq);
  //for a non-root node to add forwarded record <source id, source sequence, next-hop id>
  command void addForwarded(am_addr_t src, uint8_t seq, am_addr_t next);
  //return the trust level of a node
  command uint16_t getTrust(am_addr_t id);
}
```

Fig. 4. *TrustManager* component.

A root broadcasts two types of delivery failure record: at most three packets of significant undelivered intervals for individual origins and at most two packets of the id's of the origins without any record in the current period. For each origin, at most three significant undelivered intervals are broadcast. For a non-root node, considering the processing and memory usage overhead, the record table keeps the forwarded message intervals for up to 20 source nodes, with up to 5 non-overlapped intervals for each individual origin. Our later experiments verify that such size limit of the table on a non-root node produces a resilient TARF with moderate overhead. The record table on a node keeps adding entries for new origins until it is full.

With our current implementation, a valid trust value is an integer between 0 and 100, and any node is assigned an initial trust value of 50. The weigh parameters are: $w_{upgrade} = 0.1$, $w_{degrade} = 0.3$. The trust table of a

non-root node node keeps the trust level for up to 10 neighbors. Considering that an attacker may present multiple fake id's, the implementation evicts entries with a trust level close to the initial trust of any node. Such eviction policy is to ensure that the trust table remembers those neighbors with high trust and low trust; any other neighbor not in this table is deemed to have the initial trust value of 50.

5.2 Incorporation of TARF into Existing Protocols

To demonstrate how this TARF implementation can be integrated into the exiting protocols with the least effort, we incorporated TARF into a collection tree routing protocol (CTP) [32]. The CTP protocol is efficient, robust, and reliable in a network with highly dynamic link topology. It quantifies link quality estimation in order to choose a next-hop node. The software platform is TinyOS 2.x. To perform the integration, after proper interface wiring, invoke the `TrustControl.start` command to enable the trust evaluation; call the `Record.addForwarded` command for a non-root node to add forwarded record once a data packet has been forwarded; call the `Record.addDelivered` command for a root to add delivered record once a data packet has been received by the root. Finally, inside the CTP's task to update the routing path, call the `Record.getTrust` command to retrieve the trust level of each next-hop candidate; an algorithm taking trust into routing consideration is executed to decide the new next-hop neighbor (see Figure 5).

Similar to the original CTP's implementation, the implementation of this new protocol decides the next-hop neighbor for a node with two steps (see Figure 5): Step 1 traverses the neighborhood table for an optimal candidate for the next hop; Step 2 decides whether to switch from the current next-hop node to the optimal candidate found. For Step 1, as in the CTP implementation, a node would not consider those links congested, likely to cause a loop, or having a poor quality lower than a certain threshold. This new implementation prefers those candidates with higher trust levels; in certain circumstances, regardless of the link quality, the rules deems a neighbor with a much higher trust level to be a better candidate (see Figure 5). The preference of highly trustable candidates is based on the following consideration: on the one hand, it creates the least chance for an adversary to misguide other nodes into a wrong routing path by forging the identity of an attractive node such as a root; on the other hand, forwarding data packets to a candidate with a low trust level would result in many unsuccessful link-level transmission attempts, thus leading to much re-transmission and a potential waste of energy. When the network *throughput* becomes low and a node has a list of low-trust neighbors, the node will exclusively use the trust as the criterion to evaluate those neighbors for routing decisions. As show in Figure 5, it uses trust/cost as a criteria only when the candidate has a trust level above certain threshold. The reason is, the sole trust/cost criteria could be exploited by an adversary replaying the routing information from a base station and thus pretending to be an extremely attractive node. As for Step 2, compared to the CTP implementation, we add two more circumstances when a node decides to switch to the optimal candidate found at Step 1: that candidate has a higher trust level, or the current next-hop neighbor has a too low trust level.

```
//Step 1. traverse the neighborhood table for an optimal candidate for the next hop
optimal_candidate = NULL
//the cost of routing via the optimal candidate provided by the existing protocol, initially infinity
optimal_cost = MAX_COST
//the trust level of the optimal candidate, initially 0
optimal_trust = MIN_TRUST
for each candidate in the neighborhood table
    if link is congested, or may cause a loop, or does not pass quality threshold
        continue
    better = false
    if candidate.trust >= optimal_trust && candidate.cost < optimal_cost
        better = true
    //prefer trustworthy candidates
    if candidate.trust >= TRUST_THRESHOLD && optimal_trust < TRUST_THRESHOLD
        better = true
    if candidate.trust >= ESSENTIAL_DIFFERENCE_THRESHOLD + optimal_trust
        better = true
    //effective when all nodes have low trust due to network change or poor connectivity
    if candidate.trust >= 3 * optimal_trust / 2
        better = true
    //add restriction of trust level requirement
    if candidate.trust >= TRUST_THRESHOLD && candidate.trust / candidate.cost >
optimal_trust / optimal_cost
        better = true
    if better == true
        optimal_candidate = candidate
        optimal_cost = candidate.cost
        optimal_trust = candidate.trust

//Step 2. decide whether to switch from the current next-hop node to the optimal candidate found:
if optimal_trust >= currentNextHop.trust                                    \
 || currentNextHop.trust <= TRUST_THRESHOLD                                 \
 || current link is congested and switching is not likely to cause loops    \
 || optimal_cost + NEXTHOP_SWITCH_THRESHOLD < currentNextHop.cost           \
    currentNextHop = optimal_candidate
```

Fig. 5. Routing decision incorporating trust management.

This new implementation integrating TARF requires moderate program storage and memory usage. We implemented a typical TinyOS data collection application, MultihopOscilloscope, based on this new protocol. The MultihopOscilloscope application, with certain modified sensing parameters for our later evaluation purpose, periodically makes sensing samples and sends out the sensed data to a root via multiple routing hops. Originally, MultihopOscilloscope uses CTP as its routing protocol. Now, we list the ROM size and RAM size requirement of both implementation of MultihopOscilloscope on non-root Telosb motes in Table 1. The enabling of TARF in MultihopOscilloscope increases the size of ROM by around 1.3KB and the size of memory by around 1.2KB.

TABLE 1
Size comparison of MultihopOscilloscope implementation

Protocol	ROM (bytes)	RAM (bytes)
CTP	31164	3579
TARF-enabled CTP	34290	4767

5.3 Empirical Evaluation on Motelab

We evaluated the performance of TARF against a combined *sinkhole* and *wormhole* attack on Motelab [36] at Harvard University. 184 TMote Sky sensor motes were deployed across many rooms at three floors in the department building (see Figure 6), with two to four motes in most rooms. Around 97 nodes functioned properly while the rest were either removed or disabled. Each mote has a 2.4GHz Chipcon CC2420 radio with an indoor range of approximately 100 meters. In Figure 6, the thin green lines indicate the direct (one-hop) wireless connection between motes. Certain wireless connection also exists between nodes from different floors.

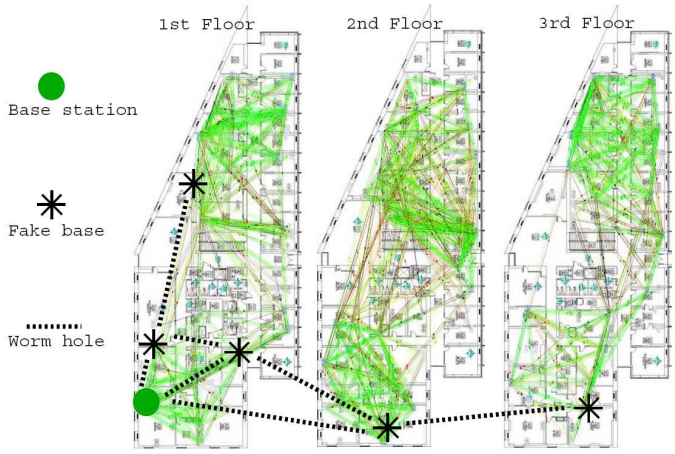


Fig. 6. Connectivity map of Motelab (not including the inter-floor connectivity), adapted from Motelab[Motelab].

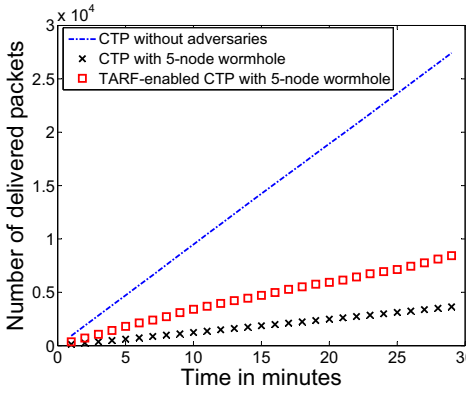
We developed a simple data collection application in TinyOS 2.x that sends a data packet every five seconds to a base station node (root) via multi-hop. This application was executed on 91 functioning non-root nodes on Motelab. For comparison, we used CTP and the TARF-enabled CTP implementation as the routing protocols for the data collection program separately. The TARF-enabled CTP has a TARF period of 30 seconds. We conducted an attack with five fake base stations that formed a *wormhole*. As in Figure 6, whenever the base station sent out any packet, three fake base stations which overheard that packet replayed the complete packet without changing any content including the node id. Other fake base stations overhearing that replayed packet would also replay the same packet. Each fake base station essentially launched a *sinkhole* attack. Note that there is a distinction between such malicious replay and the forwarding when a well-behaved node receives a broadcast from the base station. When a well-behaved node forwards a broadcast packet from the base station, it will include its own id in the packet so that its receivers will not recognize the forwarder as a base station. We conducted the first experiment by uploading the program with the CTP protocol onto 91 motes (not including those 5 selected motes as fake bases in later experiments), and no attack was involved here. Then, in another experiment, in addition to programming those 91 motes with CTP, we also programmed the five fake base stations so that they stole the id the base station through replaying. In the last experiment, we programmed those 91 motes with the TARF-enabled CTP, and programmed the five fake base stations as in the second experiment. Each of our programs run for 30 minutes.

As illustrated in Figure 7(a), the existence of the five *wormhole* attackers greatly degraded the performance of CTP: the number of the delivered data packets in the case of CTP with the five-node *wormhole* is no more than 14% that in the case of CTP without adversaries. The TARF-enabled CTP succeeded in bringing an immense improvement over CTP in the presence of the five-node *wormhole*, almost doubling the *throughput*. That improvement did not show any sign of slowing down as time elapsed. The number of nodes from each floor that delivered at least one data packet in each six-minute sub-period is plotted in Figure 7(a), 7(b) and 7(c) separately. On each floor, without any adversary, at least 24 CTP nodes were able to find a successful route in each six minute. However, with the five fake base stations in the *wormhole*, the number of CTP nodes that could find a successful route goes down to 9 for the first floor; it decreases to no more than 4 for the second floor; as the worst impact, none of the nodes on the third floor ever found a successful route. A further look at the data showed that all the nine nodes from the first floor with successful delivery record were all close to the real base station. The CTP nodes relatively far away from the base station, such as those on the second and the third floor, had little luck in making good routing decisions. When TARF was enabled on each node, most nodes made correct routing decisions circumventing the attackers. That improvement can be verified by the fact that the number of the TARF-enabled nodes with successful delivery record under the threat of the *wormhole* is close to that of CTP nodes with no attackers, as shown in Figure 7(a), 7(b) and 7(c).

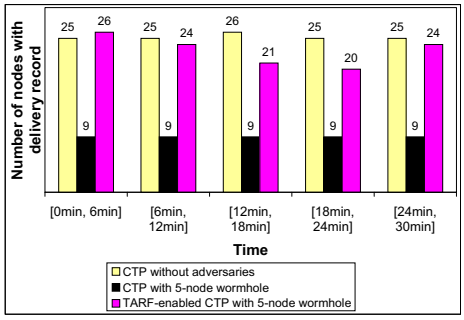
5.4 Application: Mobile Target Detection in the Presence of an Anti-Detection Mechanism

To demonstrate how TARF can be applied in networked sensing systems, we developed a proof-of-concept resilient application of target detection. This application relies on a deployed wireless sensor network to detect a target that could move, and to deliver the detection events to a base station via multiple hops with the TARF-enabled CTP protocol. For simplification, the target is a LEGO MINDSTORM NXT 2.0 vehicle robot equipped with a TelosB mote that sends out an AM (Active Message) packet every three seconds. A sensor node receiving such a packet from the target issues a detection report, which will be sent to the base station with the aforementioned TARF-enabled CTP protocol.

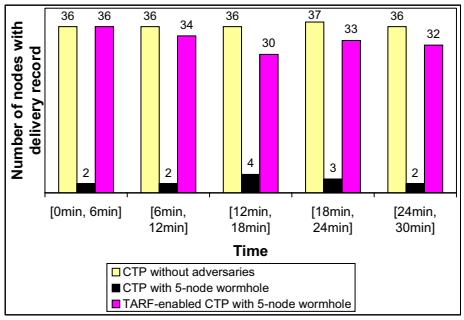
The experiment is set up within a clear floor space of 90 by 40 inches with 15 TelosB motes (see Figure 8(a)). To



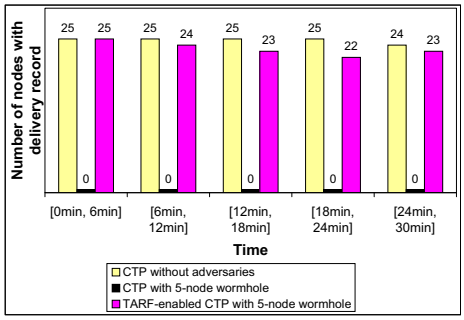
(a) All three floors.



(b) First floor.



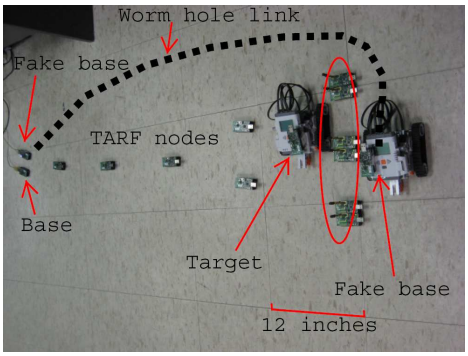
(c) Second floor.



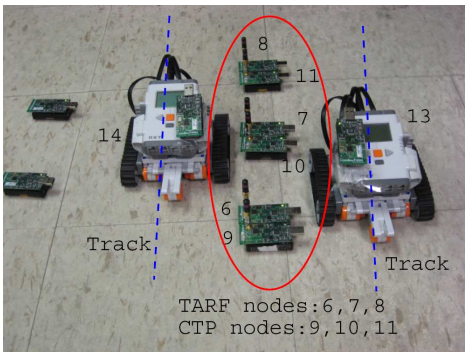
(d) Third floor.

Fig. 7. Empirical comparison of CTP and TARF-enabled CTP on Motelab: (a) number of all delivered data packets since the beginning; number of nodes on (b) the first floor, (c) the second floor and (d) the third floor that delivered at least one data packet in sub-periods.

make the multi-hop delivery necessary, the transmission power of all the Telosb motes except two fake base stations in the network is reduced through both software reduction and attenuator devices to within 30 inches. The target uses an anti-detection mechanism utilizing a fake base station close to the real base station, and another remote base station close to the target and mounted on another LEGO vehicle robot. The two fake base stations, with a transmission range of at least 100 feet, collude to form a *wormhole*: the fake base station close to the base station replays all the packets from the base station immediately; the remote fake base station, after receiving those packets, immediately replays it again. This anti-detection mechanism tricks some network nodes into sending their event reports into these fake base stations instead of the real base station. Though the fake base station close to the real base station is capable of cheating the whole network alone by itself with its powerful radio for a certain amount of time, it can be easily recognized by remote nodes as a poor next-hop candidate soon by most routing protocols based on link quality: that fake base station does not acknowledge the packets "sent" to it from remote nodes with a weak radio via a single hop since it can not really receive them. Thus, the anti-detection mechanism needs to create such a *wormhole* to replay the packets from the base station remotely.



(a) A snapshot of the network.

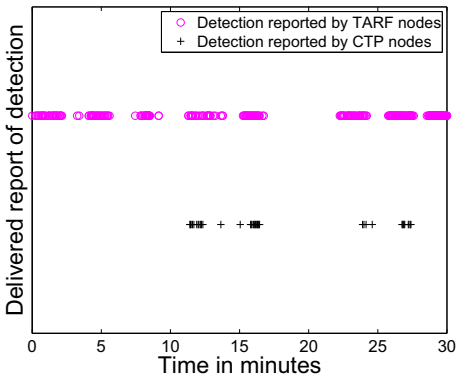


(b) A closer look.

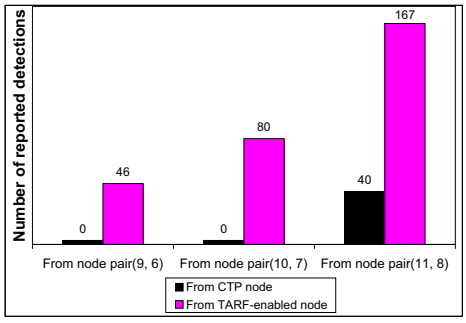
Fig. 8. Deployment of a TARF-enabled wireless sensor network to detect a moving target under the umbrella of two fake base stations in a *wormhole*.

The target node 14 and the fake base station 13 close to it move across the network along two parallel tracks of 22 inches back and forth (see Figure 8(b)); they travel

on each forward or backward path of 22 inches in around 10 minutes. The experiment lasts 30 minutes. For comparison, three nodes 9, 10 and 11 programmed with the CTP protocol are paired with another three nodes 6, 7 and 8 programmed with the TARF-enabled CTP (see Figure 8(b)); each pair of nodes are physically placed close enough. All the other nodes, except for the fake base stations and the target node, are programmed with the TARF-enabled CTP. To fairly compare the performance between CTP and the TARF-enabled CTP, we now focus on the delivered detection reports originating from these three pairs of nodes: pair (9, 6), (10, 7) and (11, 8). For the timestamp of each detection report from these six nodes, we plot a corresponding symbol: a purple circle for the nodes with the TARF-enabled CTP; a black cross for the CTP nodes. The resulting detection report is visualized in Figure 9(a). Roughly, the TARF nodes report the existence of the target seven times as often as the CTP nodes do. More specifically, as shown in Figure 9(b), in the pair (9, 6), no report from CTP node 9 is delivered while 46 reports from TARF node 6 is delivered; in the pair (10, 7), no report from CTP node 10 is delivered while 80 reports from TARF node 7 is delivered; in the pair (11, 8), 40 reports from CTP node 11 is delivered while 167 reports from TARF node 8 is delivered. Taking into account the spatial proximity between each pair of nodes, the TARF-enabled CTP achieves an enormous improvement in target detection over the original CTP.



(a) Detection report.



(b) Number of reported detections.

Fig. 9. Comparison of CTP and the TARF-enabled CTP in detecting the moving target.

The demonstration of our TARF-based target detection application implies the significance of adopting a secure routing protocol in certain critical applications. The experimental results indicate that TARF greatly enhances the security of applications involving multi-hop data delivery.

6 Related Work

We discuss more related work here in addition to the introduction in Section 1. It is generally hard to protect WSNs from *wormhole* attacks, *sinkhole* attacks and *Sybil* attacks based on identity deception. The countermeasures often requires either tight time synchronization or known geographic information [4]. FBSR[37], as a feedback-based secure routing protocol for WSNs , uses a statistics-based detection on a base station to discover potentially compromised nodes. But the claim that FBSR is resilient against *wormhole* and *Sybil* attacks is never evaluated or examined; the Keyed-OWHC-based authentication used by FBSR also causes considerable overhead. There also exists other work on trust-aware secure routing that is evaluated only through computer simulation, such as [38].

There are certain existing secure routing solutions for WSNs based on trust and reputation management; however, they rarely address the "identity theft" exploiting the replay of routing information. Two such representative solutions are ATSR [22] and TARP [23]. Neither ATSR nor TARP offers protection against the identity deception through replaying routing information. ATSR [22] is a location-based trust-aware routing solution for large WSNs. ATSR incorporates a distributed trust model utilizing both direct and indirect trust, geographical information as well as authentication to protect the WSNs from packet misforwarding, packet manipulation and acknowledgements spoofing. Another trust-aware routing protocol for WSNs is TARP [23], which exploits nodes' past routing behavior and link quality to determine efficient paths.

7 Conclusions

We have designed and implemented TARF, a robust trust-aware routing framework for WSNs, to secure multi-hop routing in dynamic WSNs against harmful attackers exploiting the replay of routing information. TARF focuses on trustworthiness and energy efficiency, which are vital to the survival of a WSN in a hostile environment. 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. Our main contributions are listed as follows. (1) Unlike previous efforts at secure routing for WSNs, TARF effectively protects WSNs from severe attacks through replaying routing information; it requires neither tight time synchronization nor known geographic information. (2) The resilience and scalability of TARF is proved through both extensive simulation and empirical evaluation with

large-scale WSNs; the evaluation involves both static and mobile settings, hostile network conditions, as well as strong attacks such as *wormhole* attacks and *Sybil* attacks. (3) We have implemented a ready-to-use TinyOS module of TARF with low overhead; as demonstrated in the paper, this TARF module can be integrated into existing routing protocols with the least effort, thus producing secure and efficient fully-functional protocols. (4) Finally, we demonstrate a proof-of-concept mobile target detection application that is built on top of TARF and is resilient in the presence of an anti-detection mechanism; that indicates the potential of TARF in WSN applications.

REFERENCES

[1] G. Zhan, W. Shi, and J. Deng, "Tarf: A trust-aware routing framework for wireless sensor networks," in *Proceeding of the 7th European Conference on Wireless Sensor Networks (EWSN'10)*, 2010.

[2] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann Publishers, 2004.

[3] A. Wood and J. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, Oct 2002.

[4] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," in *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[5] M. Jain and H. Kandwal, "A survey on complex wormhole attack in wireless ad hoc networks," in *Proceedings of International Conference on Advances in Computing, Control, and Telecommunication Technologies (ACT '09)*, 28-29 2009, pp. 555 –558.

[6] I. Krontiris, T. Giannetsos, and T. Dimitriou, "Launching a sinkhole attack in wireless sensor networks; the intruder side," in *Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications(WIMOB '08)*, 12-14 2008, pp. 526 –531.

[7] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis and defenses," in *Proc. of the 3rd International Conference on Information Processing in Sensor Networks (IPSN'04)*, Apr. 2004.

[8] L. Bai, F. Ferrese, K. Ploskina, and S. Biswas, "Performance analysis of mobile agent-based wireless sensor network," in *Proceedings of the 8th International Conference on Reliability, Maintainability and Safety (ICRMS 2009)*, 20-24 2009, pp. 16 –19.

[9] L. Zhang, Q. Wang, and X. Shu, "A mobile-agent-based middleware for wireless sensor networks data fusion," in *Proceedings of Instrumentation and Measurement Technology Conference (I2MTC '09)*, 5-7 2009, pp. 378 –383.

[10] W. Xue, J. Aiguo, and W. Sheng, "Mobile agent based moving target methods in wireless sensor networks," in *IEEE International Symposium on Communications and Information Technology (ISCIT 2005)*, vol. 1, 12-14 2005, pp. 22 – 26.

[11] J. Hee-Jin, N. Choon-Sung, J. Yi-Seok, and S. Dong-Ryeol, "A mobile agent based leach in wireless sensor networks," in *Proceedings of the 10th International Conference on Advanced Communication Technology (ICACT 2008)*, vol. 1, 17-20 2008, pp. 75 –78.

[12] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless Communications*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[13] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Proc. of ACM SenSys 2004*, Nov. 2004.

[14] A. Perrig, R. Szewczyk, W. Wen, D. Culler, and J. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks Journal (WINET)*, vol. 8, no. 5, pp. 521–534, Sep. 2002.

[15] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: securing sensor networks with public key technology," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN '04)*. New York, NY, USA: ACM, 2004, pp. 59–64.

[16] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proceedings of the 7th international conference on Information processing in sensor networks (IPSN '08)*. IEEE Computer Society, 2008, pp. 245–256.

[17] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[18] H. Safa, H. Artail, and D. Tabet, "A cluster-based trust-aware routing protocol for mobile ad hoc networks," *Wirel. Netw.*, vol. 16, no. 4, pp. 969–984, 2010.

[19] W. Gong, Z. You, D. Chen, X. Zhao, M. Gu, and K. Lam, "Trust based routing for misbehavior detection in ad hoc networks," *Journal of Networks*, vol. 5, no. 5, May 2010.

[20] Z. Yan, P. Zhang, and T. Virtanen, "Trust evaluation based security solution in ad hoc networks," in *Proceeding of the 7th Nordic Workshop on Secure IT Systems*, 2003.

[21] J. L. X. Li, M. R. Lyu, "Taodv: A trusted aodv routing protocol for mobile ad hoc networks," in *Proceedings of Aerospace Conference*, 2004.

[22] T. Zahariadis, H. Leligou, P. Karkazis, P. Trakadas, I. Papaefstathiou, C. Vangelatos, and L. Besson, "Design and implementation of a trust-aware routing protocol for large wsns," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 2, no. 3, Jul. 2010.

[23] A. Rezgui and M. Eltoweissy, "Tarp: A trust-aware routing protocol for sensor-actuator networks," in *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS 2007)*, 8-11 2007.

[24] A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Comput. Commun.*, vol. 30, pp. 2826–2841, October 2007.

[25] S. Chang, S. Shieh, W. Lin, and C. Hsieh, "An efficient broadcast authentication scheme in wireless sensor networks," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (ASIACCS '06)*. New York, NY, USA: ACM, 2006, pp. 311–320.

[26] K. Ren, W. Lou, K. Zeng, and P. Moran, "On broadcast authentication in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 11, pp. 4136 –4144, november 2007.

[27] P. De, Y. Liu, and S. K. Das, "Modeling node compromise spread in wireless sensor networks using epidemic theory," in *World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a*, 2006, pp. 7 pp. –243.

[28] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the First ACM SenSys'03*, Nov. 2003.

[29] S. Ganeriwal, L. Balzano, and M. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Trans. Sen. Netw.*, 2008.

[30] G. Zhan, W. Shi, and J. Deng, "Poster abstract: Sensortrust - a resilient trust model for wsns," in *Proceedings of the 7th International Conference on Embedded Networked Sensor Systems (SenSys'09)*, 2009.

[31] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 234–244, 1994.

[32] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*. New York, NY, USA: ACM, 2009, pp. 1–14.

[33] G. Zhan, W. Shi, and J. Deng, "Design, implementation and evaluation of tarf: A trust-aware routing framework for dynamic wsns," http://mine.cs.wayne.edu/~guoxing/tarf.pdf, Wayne State University, Tech. Rep. MIST-TR-2010-003, Oct. 2010.

[34] Q. Zheng, X. Hong, and S. Ray, "Recent advances in mobility modeling for mobile ad hoc network research," in *Proceedings of the 42nd Annual Southeast Regional Conference (ACM-SE 42)*. New York, NY, USA: ACM, 2004, pp. 70–75.

[35] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '99)*. New York, NY, USA: ACM, 1999, pp. 53–60.

[36] "Motelab," http://motelab.eecs.harvard.edu, 2005.

[37] Z. Cao, J. Hu, Z. Chen, M. Xu, and X. Zhou, "Fbsr: feedback-based secure routing protocol for wireless sensor networks," *International Journal of Pervasive Computing and Communications*, 2008.

[38] T. Ghosh, N. Pissinou, and K. Makki, "Collaborative trust-based secure routing against colluding malicious nodes in multi-hop ad hoc networks," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, Nov. 2004, pp. 224 – 231.

Guoxing Zhan Guoxing Zhan is currently a Ph.D. candidate in the Department of Computer Science at Wayne State University. He received a M.S. in Mathematics from Chinese Academy of Sciences in 2007 and another M.S in Computer Science from Wayne State University in 2009. Mr. Zhan is interested in research on participatory sensing, wireless sensor network, mobile computing, networking and systems Security, trust Management, and information processing. Several of his research papers have been presented at international conferences or published in journals. Additionally, Mr. Zhan has instructed a few computer science labs consisting of interactive short lectures and hands-on experience.

Weisong Shi Dr. Weisong Shi is an Associate Professor of Computer Science at Wayne State University. He received his B.S. from Xidian University in 1995, and Ph.D. degree from the Chinese Academy of Sciences in 2000, both in Computer Engineering. His current research focuses on computer systems, mobile and cloud computing. Dr. Shi has published more than 100 peer reviewed journal and conference papers. He is the author of the book “Performance Optimization of Software Distributed Shared Memory Systems” (High Education Press, 2004). He has served the program chairs and technical program committee members of several international conferences. He is a recipient of the NSF CAREER award, one of 100 outstanding Ph.D. dissertations (China) in 2002, Career Development Chair Award of Wayne State University in 2009, and the “Best Paper Award” of ICWE’04 and IPDPS’05.

Julia Deng Dr. Julia (Hongmei) Deng currently is a Principal Scientist at Intelligent Automation Inc. Her primary research interests include protocol design, analysis and implementation in wireless ad hoc/sensor networks, network security, information assurance, and network management. Dr Deng received her Ph.D. in the Department of Electrical Engineering from the University of Cincinnati in 2004, majoring in communications and computer networks. At IAI, she serves as the PI and leads many network and security related projects, such as secure routing for airborne Networks, network service for airborne Networks, DoS mitigation for tactical networks, trust-aware querying for sensor networks, trusted routing for sensor networks, agent-based intrusion detection system, just to name a few.