E^2TARF: Energy-efficient Trust-Aware Routing Framework for WSNs

Kamleshwar Kumar Yadav
6th Sem, M.Tech(CSE)
SJCIT Chickaballapur-562101
Kewalkamlesh07@gmail.com

Mrs. Anitha T. N.
Asso. Professor ,CSE Dept
SJCIT Chickaballapur-562101
anithareddytn72@gmail.com

Abstract— A WSN comprises battery-powered sensor 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. 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,TARF provides trustworthy and energy-efficient route. Most importantly, TARF proves effective against those harmful attacks developed out of identity deception.

I. 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. .

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".

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 or attempt to exclude unauthorized participation by encrypting data and authenticating packets. 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.

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.*We start by stating the design considerations of TARF. Then we elaborate the design of TARF , including the routing procedure as well as the *EnergyWatcher* and *TrustManager* components

II. DESIGN CONSIDERATIONS

Authentication: TARF requires that the packets are properly authenticated, especially the broadcast packets from the base station. TARF may use TrustManager and the received broadcast packets about delivery information to choose trustworthy path by circumventing compromised nodes. Without being able to physically capturing the base station. 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.

Energy Efficiency: Data transmission accounts for a major portion of the energy consumption. We evaluate energy efficiency by the average energy cost to success –fully deliver a unit-sized data packet from a source node to the base station. 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.*

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. *Through-put* reflects how efficiently the network is collecting and delivering data.

III. DESIGN OF E^{2}TARF

E^{2}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. TARF is also energy-efficient, highly scalable, and well adaptable.

Neighbor For a node N, a neighbor 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.

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 *Trust - Manager*, run on the node (Figure 1). *EnergyWatcher* is responsible for recording the energy cost for each known neighbor 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 by *EnergyWatcher*. Such an energy cost report also serves as the input of its receivers.

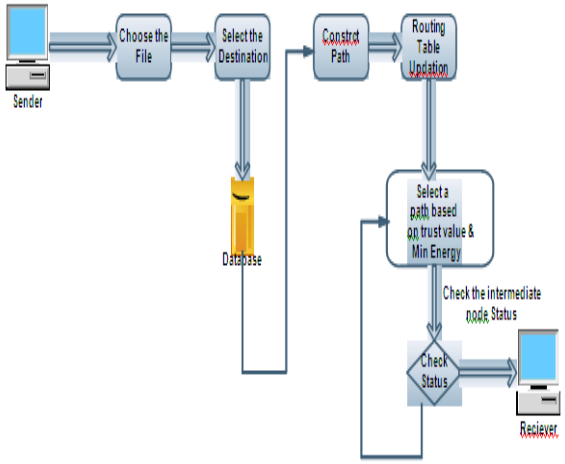

FIGURE 1:*Energy-Watcher* and *TrustManager* on the node keep track of related events to record the energy cost and the trust level values of its neighbors.

Routing Procedure: 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. 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.

EnergyWatcher:

we describe how a node N's *EnergyWatcher* computes the energy cost ENb for its neighbor b in N's neighborhood table and how N decides its own energy cost EN. Before going further, we will clarify some notations. ENb 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 retrains-mission may occur until the acknowledgement is received or the number of re-transmissions reaches a certain threshold. The cost caused by one-hop retransmissions should be included when computing ENb. Suppose N decides that A should be its next-hop node after comparing energy cost and trust level. Then N's energy cost is EN = ENA. Denote EN→b as the average energy cost of successfully delivering a data packet from N to its neighbor b with one hop. Note that the retransmission cost needs to be considered. With the above notations, it is straightforward to establish the following relation:

$$ENb = EN \rightarrow b + Eb$$

Since each known neighbor b of N is supposed to broadcast its own energy cost Eb to N, to compute ENb, N still needs to know the value EN→b.

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, TNb 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 node id of a source node, a forwarded sequence interval [a, b] with a significant length> 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 Told 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.

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

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.

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

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 Delivery Ratio. Then N's *TrustManager* updates its next-hop node b's trust.

IV. ANALYSIS ON ENERGYWATCHER AND TRUSTMANAGER

Node N relies on its *EnergyWatcher* and *TrustManager* to select an optimal neighbor as its nexthop 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 note worthy 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. Figure 2 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

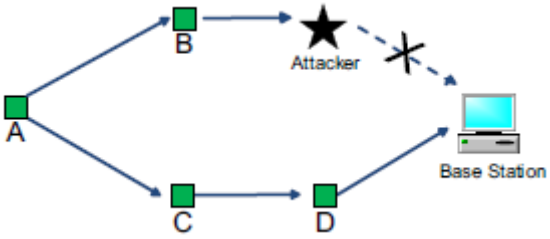

Figure. 2. An example to how *TrustManager* works.

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, 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.

V. CONCLUSIONS

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. TARF effectively protects WSNs from severe attacks through replaying routing information; it requires neither tight time synchronization nor known geographic information. 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. TARF module can be integrated into existing routing protocols with the least effort, thus producing
secure and efficient fully-functional protocols.

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 Con-ference 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*

[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*.

[9] 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.

[10] 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

[11] 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.