An Efficient Tarf for Detecting Different Attacks in Networks

K.Anusha, (M.Tech), Chadalawada Ramanamma Engineering College
Mr.J.Ravi Kumar, Assistant Professor, Chadalawada Ramanamma Engineering College

Abstract- In networks, where privacy, especially the location privacy of anonymous vehicles is highly concerned, anonymous verification of vehicles is indispensable. Consequently, an attacker who succeeds in forging multiple hostile identifies can easily launch a Sybil attack, gaining a disproportionately large influence. In this paper, 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 Tiny OS; 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

Over the past two decades, vehicular networks have been emerging as a cornerstone of the next-generation Intelligent Transportation Systems (ITSs), contributing to safer and more efficient roads by providing timely information to drivers and concerned authorities. In vehicular networks, moving vehicles are enabled to communicate with each other via intervehicle communications as well as with road-side units (RSUs) in vicinity via roadside-to-vehicle communications. In urban vehicular networks where the privacy, especially the location privacy of vehicles should be guaranteed [1], [2], vehicles need to be verified in an anonymous manner. A wide spectrum of applications in such a network relies on collaboration and information aggregation among participating vehicles. Without identities of participants, such applications are vulnerable to the Sybil attack where a malicious vehicle masquerades as multiple identities [3], overwhelmingly influencing the result. The consequence of Sybil attackhappening in vehicular networks can be vital. For example, in safety-related applications such as hazard warning, collision avoidance, and passing assistance, biased results caused by a Sybil attack can lead to severe car accidents. Therefore, it is of great importance to detect Sybil attacks from the very beginning of their happening.

Detecting Sybil attacks in urban vehicular networks, however, is very challenging. First, vehicles are anonymous. There are no chains of trust linking claimed identities to real vehicles. Second, location privacy of vehicles is of great concern. Location information of vehicles can be very confidential. For example, it can be inferred that the driver of a vehicle may be sick from knowing the vehicle is parking at a hospital. It is inhibitive to enforce a one-to-one correspondence between claimed identities to real vehicles by verifying the physical presence of a vehicle at a particular place and time. Third, conversations between vehicles are very short. Due to high mobility of vehicles, a moving vehicle can have only several seconds [4] to communicate with another occasionally encountered vehicle. It is difficult to establish certain trustworthiness among communicating vehicles in such a short time. This makes it easy for a malicious vehicle to generate a hostile identity but very hard for others to validate. Furthermore, short conversations among vehicles call for online Sybil attack detection. The detection scheme fails if a Sybil attack is detected after the attack has terminated.

In this paper, we propose a novel Sybil attack detection scheme Footprint, using the trajectories of vehicles for identification while

still preserving the anonymity and location privacy of vehicles. Specifically, in Footprint, when a vehicle encounters an RSU, upon request, the RSU issues an authorized message for this vehicle as the proof of its presence at this RSU and time. Intuitively, authorized messages can be utilized to identify vehicles since vehicles located at different areas can get different authorized messages. However, directly using authorized messages will leak location privacy of vehicles because knowing an authorized message of a vehicle signed by a particular RSU is equivalent to knowing the fact that the vehicle has showed up near that RSU at that time. In Footprint, we design a location-hidden authorized message generation scheme for two purposes. First, RSU signatures on messages are signer-ambiguous which means an RSU is anonymous when signing a message. In this way, the RSU location information is concealed from the final authorized message. Second, authorized messages are temporarily linkable which means two authorized messages issued from the same RSU are recognizable if and only if they are issued within the same period of time. Thus, authorized messages can be used for identification of vehicles even without knowing the specific RSUs who signed these messages. With the temporal limitation on the linkability of two authorized messages, authorized messages used for longterm identification are prohibited. Therefore, using authorized messages for identification of vehicles will not harm anonymity of vehicles.

2. System Model and Assumption

In vehicular networks, a moving vehicle can communicate with other neighboring vehicles or RSUs via intervehicle communications and roadside-to-vehicle communications. Fig. 1 illustrates the architecture of the system model, which consists of three interactive components:

- RSUs: can be deployed at intersections or any area of interest (e.g., bus stations and parking lot Entrances). A typical RSU also functions as a wireless AP (e.g., IEEE 802.11x) which provides wireless access to users within its coverage. RSUs are interconnected (e.g., by a dedicated network or through the Internet via cheap ADSL connections) forming a RSU backbone network.

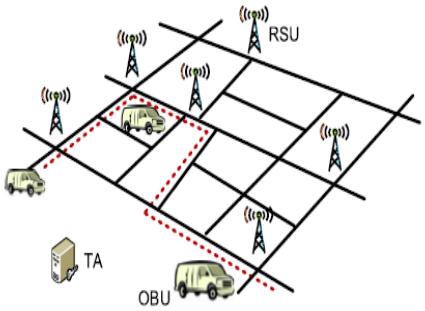


Fig. 1. An illustration of the system model, where the dash line indicates the travel route of a vehicle. As the vehicle traverses the area, it will encounter multiple RSUs, typically deployed at intersections.

- On-board units (OBUs): are installed on vehicles. A typical OBU can equip with a cheap GPS receiver and a short-range wireless communication module (e.g., DSRC IEEE 802.11p [20]). A vehicle equipped with an OBU can communicate with an RSU or with other vehicles in vicinity via wireless connections.For simplicity, we simply refer to a vehicle as a vehicle equipped with an OBU in the rest of this paper. A vehicle can be malicious if it is an attacker or compromised by an attacker.
- Trust authority: is responsible for the system initialization and RSU management. The TA is also connected to the RSU backbone network. Note that the TA does not serve vehicles for any certification purpose in Footprint. A vehicle can claim as many arbitrary identities as it needs.

2.1 Design Goals

The design of a Sybil attack detection scheme in urban vehicular networks should achieve three goals:

1. Location privacy preservation: a particular vehicle would not like to expose its location information to other vehicles and RSUs as well since such information can be confidential. The detection scheme should prevent the location information of vehicles from being leaked.

2. Online detection: when a Sybil attack is launched, the detection scheme should react before the attack has terminated. Otherwise, the attacker could already achieve its purpose.

3. Independent detection: the essence of Sybil attack happening is that the decision is made based on
group negotiations. To eliminate the possibility that a Sybil attack is launched against the detection itself,
the detection should be conducted independently by the verifier without collaboration with others.

3. System Design

3.1 Overview

In general, Footprint integrates three elegant techniques namely, infrastructure construction, location-hidden trajectory generation, and Sybil attack detection.

More specifically, we adopt an incremental methodology to deploy RSUs. In the end, a limited number of available RSUs can achieve the maximum service coverage in terms of served traffic amount as well as good fairness in terms of geographical distribution. After the deployment of RSUs, a vehicle can require authorized messages from each RSU it passes by as a proof of its presence there. We adopt an event-oriented linkable ring signature scheme [24] for RSUs to issue authorized messages for vehicles. Such authorized messages are location hidden which refers to that RSU signatures are signer ambiguous and the authorized messages are temporarily linkable. Furthermore, a set of consecutive authorized messages issued for a vehicle are tightly chained together to form a location-hidden trajectory of the vehicle, which will be utilized for identifying this vehicle in future conversations. During a conversation which is initialized by a vehicle or an RSU, called a conversation holder, a participating vehicle should provide its trajectory for verification. With the trajectories sent from all participating vehicles, the conversation holder can conduct online Sybil attack detection according to the similarity relationship between each pair of trajectories. Among all trajectories, Sybil trajectories forged from the same attacker are bound to gather within the same "community."

3.2 Infrasttructre Construction

3.2.1 *RSU Development :*

In Footprint, vehicles require authorized messages issued from RSUs to form trajectories, which should be statically installed as the infrastructure. When considering the deployment of RSUs, two practical questions are essential, i.e., where to install RSUs in the city and how many of them are sufficient?

A simple solution is to deploy RSUs at all intersections. This can result fine trajectories with a sufficient number of authorized messages which will facilitate the recognition of a vehicle. However, deploying such a huge number of RSUs in one time is prohibitive due to the high cost.

In contrast, we take an incremental deployment strategy in Footprint, considering the tradeoff between minimizing the number of RSUs and maximizing the coverage of traffic.Specifically, in the early developing stage with a limited number of RSUs, an intersection is chosen if it satisfies two requirements: first, it is geographically at least certain distance far away from all other RSU-equipped intersections;second, it has the maximum traffic volume among all rest intersections without RSUs.

3.2.2 System Intialization

After completing RSU deployment, in order to function properly, the system first needs to be initialized. The initialization process includes three steps:

1. Setting up TA: the TA first chooses a set of public parameters required for the ring signature scheme which is used for RSUs to sign messages and establishes a pair of public/private key pair (K^{pub}_{TA};K^{pri}_{TA}) as well. The public key of the TAK^{pub}_{TA} can be obtained by all RSUs and vehicles in the system through a secure channel. It is used to verify whether a message is authorized by theTA (see Appendix C, available

in the online supplemental material, for the detailed initialization process).

2. Setting up RSUs: when a new RSU Rk is added to the system, the TA issues a pair of public/private key pair (K^{pub}_{TA};K^{pri}_{TA})I or Rk and sends the public parameters to Rk as well. After all RSUs are registered in the system, the Public Key List (PKL) of all RSUs is broadcasted to all RSUs from the TA via the RSU backbone network. In addition, the IP addresses of its neighboring RSUs of Rk are also notified to Rk. Note that all messages sent from the TA are authorized by the TA using its private key K^{pr}_{iTA}.

3. Setting up vehicles: For a vehicle to join in the system, it only needs to get the PKL of all RSUs and the public parameters. It can get such information when encountering any RSU or a vehicle with the information. After that, it can construct its own trajectories in the system.

3.3 Generating Location Hidden Trajectory

3.3.1.1 Authorization Message Generation

In order to be location hidden, authorized messages issued for vehicles from an RSU should possess two properties, i.e., signer ambiguous and temporarily linkable. The signerambiguous property means the RSU should not use a dedicated identity to sign messages. The temporarily linkable property requires two authorized messages are recognizable if and only if they are generated by the same RSU within the same given period of time. Otherwise, a long-term linkability of authorized messages used for identification eventually has the same effect as using a dedicated identity for vehicles.

authorized messages may have the same link tag. In this case, it is hard to tell whether these messages belong to different vehicles. With the independent mobility assumption, as two vehicles move along, the probability for the pair of vehicles having exactly the same trajectories is slim. Therefore, it is feasible to use trajectories to exclusively represent corresponding vehicles as long as those trajectories are sufficiently long. With authorized messages, a straightforward method for a vehicle to present its trajectory is to sort all its authorized messages into a sequence according to time.Thus, in future conversations, the vehicle can use this sequence of authorized messages to identify itself.

In this paper, we demonstrate one possible implementation of a location-hidden authorized message generation scheme using linkable ring signature [21]. Linkable ring signature is signer-ambiguous and signatures are linkable (i.e., two signatures can be linked if and only if they are issued by the same signer) as well. Particularly, we choose the linkable ring signature scheme introduced by Dodis et al. [22] and Tsang and Wei [23] for two reasons: first, it has been proved to be secure; second, it has constant signature size.To meet the requirement of temporarily linkable property, we extend the scheme to support the event-orient linkability property [24] which guarantees that any two signatures are linkable if and only if they are signed based on the same event by the same RSU.

3.3.2 Message Verification

As the proof that a vehicle v_i was present near certain RSU R_k at certain time, an authorized message issued for v_i can be verified by any entity (e.g., a vehicle or an RSU) in the system.

In the case that an entity needs to verify v_i, v_i will sign on an authorized message $\mathcal{M} \parallel S_{R_k}(\mathcal{M})$ generated by RSU R_k using $K^{pri}_{v_{i,j}}$ and then send

$$L_{R_k} = \mathcal{M} \parallel S_{R_k}(\mathcal{M}) \parallel S_{K^{pri}_{vi,j}}(\mathcal{M} \parallel S_{R_k}(\mathcal{M}))$$

3.3.3 Trajectory Encoded Message

Intuitively, an authorized message issued from an RSU can be used to identify a vehicle. However, it is often the case that two or more

3.4 Sybil Attack Detection

During a conversation, upon request from the conversation holder, all participating vehicles provide their trajectoryembedded authorized messages issued within specified event for identification. With submitted messages, the conversation holder verifies each trajectory and refuses those vehicles that fail the message verification. After that, the conversation holder conducts online Sybil attack detection before further proceeding with the conversation.

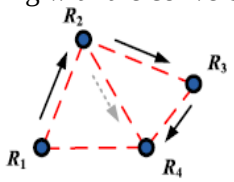


Fig. 2. RSU neighboring relationship and the freedom of trajectory generation can facilitate Sybil trajectory generation. In the above figure, neighboring RSUs (denoted by dots) are connected with dash line. The solid arrows indicate the actual sequence of RSUs a malicious meet and the dash arrow presents a possible forged trajectory.

3.4.1 Problem Definition

Recall that, in Footprint, vehicles have wide freedom to create their trajectories. For example, a vehicle is allowed to request multiple authorized messages from an RSU using different temporary key pairs. Thus, a vehicle can use different authorized messages for different conversations.

This capability, however, can be leveraged by a malicious vehicle that tries to launch a Sybil attack by using multiple different messages in a single conversation. We define the Sybil attack detection problem as: Given a set of trajectoryembedded authorized messages within an event, how can the conversation holder recognize real vehicles and Sybil ones? The online Sybil attack problem is hard due to three following factors:

First, authorized messages generated for different vehicles are asynchronous. The rationale of using trajectories to represent vehicles is based on the fact that a vehicle cannot present itself at different locations at the same time. The asynchrony of messages makes the judgment directly based on this fact impractical.

Second, authorized messages are temporarily linkable,which means there is no invariable mapping between an RSU signature and the real RSU who signed this signature. Thus, no distance information is available between two RSUs enclosed in any two signatures. This makes the problem even harder since one cannot utilize the time difference between two authorized messages and the distance between the pair of corresponding RSUs to infer whether two messages belong to two distinct vehicles.

Last, a malicious vehicle can abuse the freedom of trajectory generation and the neighbor relationship among RSUs to generate elaborately designed trajectories. For example, in Fig. 2, an attacker can legally generate multiple trajectories which appear different from each other even under a very simple RSU topology. Assume the real path of the attacker is {R1;R2;R3;R4} (indicated by solid arrows). It can start a new trajectory at any RSU by using a different temporary key pair. Therefore, besides the trajectory {R1;R2;R3;R4} trajectories like {R1;R2;R3}, {R2;R3;R4},{R1;R2}; {R2;R3}, {R3;R4}, {R1}, {R2}, {R3} and {R4} are all legitimate. In addition, knowing the neighboring relationship of R2 andR4, the attacker can generate forged trajectories like {R1;R2;R4}, R1;R4, and R2;R4 (indicated by the dash arrow). Note that the attacker cannot generate a trajectory like {R1;R3} because R1 is not a neighbor of R3.In the case of this example, R3 only expects signatures signed by R2 and R4.

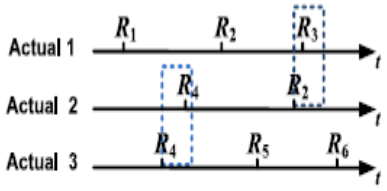


Fig. 3. Checking for distinct trajectories by using a check window (denoted as the box of dotted line) and counting the total number of different RSUs contained in a pair of trajectories.

3.4.2 Relationship and among startagies

We first thoroughly examine the characteristics of forged and actual trajectories.

Features of forged trajectories. Although a malicious vehicle can submit multiple forged trajectories to a conversation holder, these trajectories satisfy two facts. First, a forged trajectory is a proper subset of the actual trajectory. For example, in Fig. 2, all forged trajectories are fully contained in the actual trajectory. Second, any two forged trajectories cannot have two distinct RSUs at the same time. It is true because otherwise the malicious vehicle would appear at two locations at the same time.

Features of actual trajectories. Despite the asynchrony and temporarily linkable properties of uthorized messages, there are two basic facts that can be exploited to judge whether two trajectories are from two actual vehicles. First, it is very hard, if not impossible, for a single vehicle to traverse between a pair of RSUs shorter than a time limit. We define such a time limit as traverse time limit. Second, within a limited time period, the total number of RSUs traversed by a single vehicle is less than a limit.

We define such a limit as trajectory length limit. Based on these features, we first conduct an exclusion test, examining whether two trajectories are distinct. There are two cases where a pair of trajectories can pass the test (positive test). In the first case, there are two distinct RSUs appearing within a sliding time window (called check window) when checking two trajectories. We can set the size of the check window equal to the traverse time limit. For example, in Fig. 3, trajectories T 1 and T 2 are distinct since there exists a pair of different RSUs within the check window (denoted by the box of dash line), i.e., R2 and R3. In the second case, the number of RSUs contained in the merged RSU sequence of two trajectories is larger than the trajectory length limit. We merge a pair of trajectories into

$$(T_1, T_2) = \begin{cases} -1 & positive\ test \\ \dfrac{|T_1 \cap T_2|}{Min\{|T_1|, |T_2|\}} & negative\ test, \end{cases}$$

4. Proposed Algorithm

4.1 TARF

TARF secures the multihop routing in WSNs against intruders misdirecting the multihop 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.

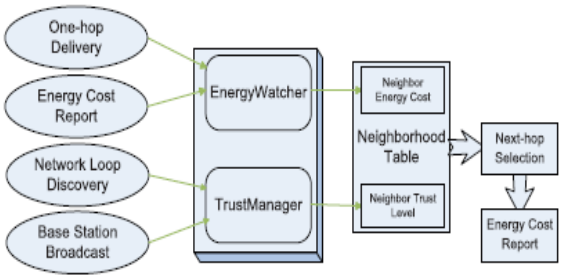
Energy cost. For a node N, the energy cost of a neighbor is the average energy cost to successfully deliver a unitsized data packet with this neighbor as its next-hop node, from N to the base station. That energy cost is denoted as E in this paper.

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 TrustManager, run on the node

EnergyWatcher is responsible for recording the energy cost for each known neighbor, based on N's observation of onehop 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.



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.

4.2 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

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

4.2.1 Structre 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 thosewithout 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 (Sections 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 toassist 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.

4.2.2 Route Selectrion

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 tradeoff between T_{Nb} and E_{Nb}/T_{Nb} E_{Nb} and T_{Nb} being b's energy cost and trust level value in the neighborhood table,respectively, (see Sections 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; 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, Enb T_{Nb} combines the trustworthiness and energy cost. However, the metric 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 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 tradeoff between T_{Nb} and Enb T_{Nb}

4.3 Energy Watcher

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 retransmission may occur until the acknowledgment is received or the number of retransmissions Reaches a certain threshold. The cost caused by onehop retransmissions 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 $E_N=E_{NA}$

$$\mathbf{E_{Nb} = E_{N->b} + E_b}$$

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 $E_{N->b}$, 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 psucc of being acknowledged, we first compute the average number of one-hop sendings needed before the acknowledgment is received as follows:

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

Denote Eunit as the energy cost for node N to send a unitsized data packet once regardless of whether it is received or not. Then, we have

$E_{Nb} = E_{unit}/P_{succ} + E_b$

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

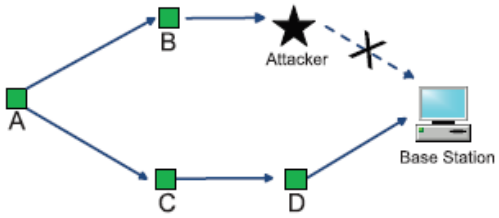
4.4 Analysis of Energy Watcher

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:

Second, TrustManager identities the low trustworthiness of various attackers misdirecting the multihop 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 evaluationresults 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.



An example to illustrate how *TrustManager* works.

5. Screenshots

Fig 1 : Client

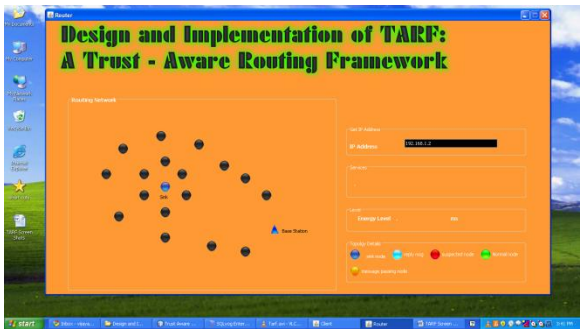


Fig 2 :Router

Fig 3 : Trust Aware Routing

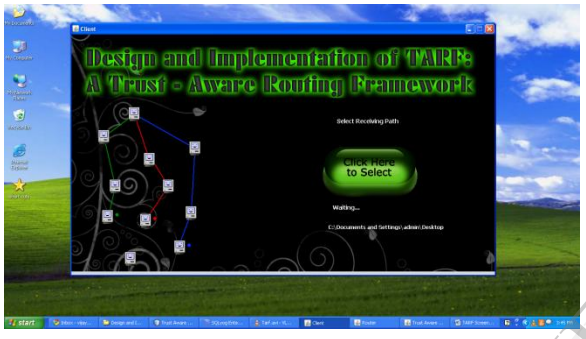


Fig 4 : Client

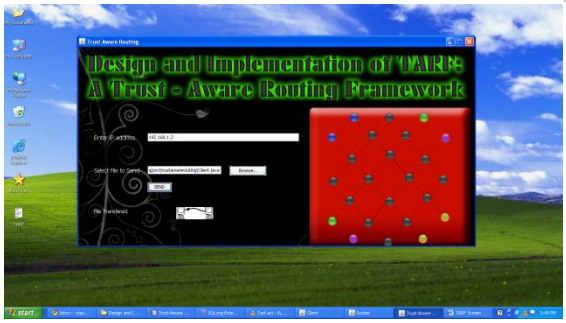


Fig 5 : Trust Aware routing

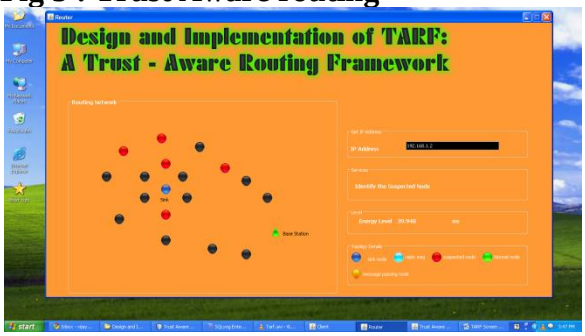


Fig 6: Router

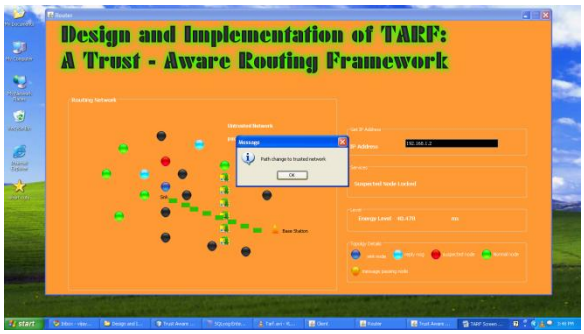


Fig 7: Router

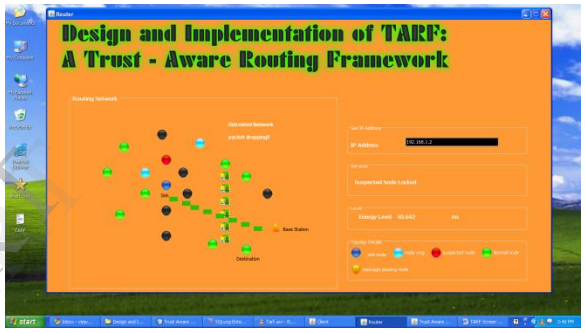


Fig 8 : Router

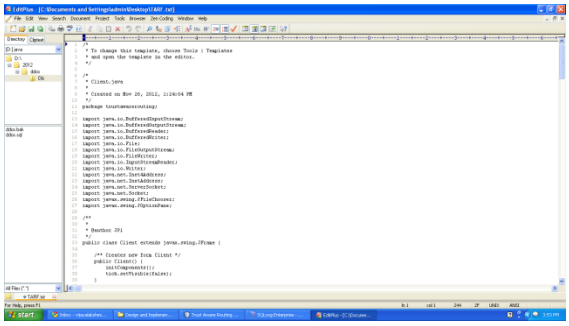


Fig 9 : Resultant File

6. Conclusion

In this paper, we have developed a Sybil attack detection scheme Footprint for urban vehicular networks. Consecutive authorized messages obtained by an anonymous vehicle from RSUs form a trajectory to identify the corresponding vehicle. Location privacy of vehicles ispreserved by realizing a location- idden signature scheme.
We have designed and implemented TARF, a robust trustaware routing framework for WSNs, to secure multihop 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 are 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 antidetection 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," Proc. Seventh European Conf. Wireless Sensor Networks (EWSN '10), 2010.
[2] F. Zhao and L. Guibas, Wireless Sensor Networks: An Information Processing Approach. Morgan Kaufmann, 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," Proc. First IEEE Int'l Workshop Sensor Network Protocols and Applications, 2003.
[5] M. Jain and H. Kandwal, "A Survey on Complex Wormhole Attack in Wireless Ad Hoc Networks," Proc. Int'l Conf. Advances in Computing, Control, and Telecomm. Technologies (ACT '09), pp. 555-558, 2009.
[6] I. Krontiris, T. Giannetsos, and T. Dimitriou, "Launching a Sinkhole Attack in Wireless Sensor Networks; The Intruder Side," Proc. IEEE Int'l Conf. Wireless and Mobile Computing, Networking and Comm. (WIMOB '08), pp. 526-531, 2008.