Secure Trust Metadata Management For Mobile Ad-Hoc Networks [†]

Vivek Natarajan[1], Yi Yang[2], and Sencun Zhu[1]

[1] Department of Computer Science and Engineering
Pennsylvania State University
{vnataraj,szhu}@cse.psu.edu
[2] Department of Electrical Engineering and Computer Science
Catholic University of America
yangy@cua.edu

Abstract. A trust management framework is useful to ensure proper functioning of a mobile ad-hoc network (MANET). Trust metadata created by individual nodes, based on their observation of the behavior of other nodes in their vicinity, is required to be accessible to a trust authority (TA) (e.g., the network administrator) for prompt decision making (e.g., revoking malicious nodes). In this work, for security and scalability reasons, we propose a secure semantics-aware trust metadata management scheme to partition and store an information network of trust metadata of nodes in a MANET. That is, trust metadata is securely propagated to and stored at certain geographic locations inside the network itself, based on its semantics. The TA can send queries of various types in the network to obtain the trust metadata of its interest. This scheme is robust to several security attacks that attempt to disrupt the availability of trust metadata in the network. Our analysis shows that the proposed scheme provides desirable security and functionality properties with low query overhead.

Keywords: Mobile Ad-Hoc Network ; Trust Metadata Management ; Semantics-Aware ; Attribute-Based Encryption

1 Introduction

A trust management framework [1–5] is useful to ensure proper functioning of a mobile ad-hoc network (MANET). Nodes in a MANET generally lack hardware support for tamper resistance. Thus, an adversary could compromise some nodes and program them to display malicious behavior. In order to address malicious behavior, trust metadata (i.e., structural data regarding the trustiness of nodes) could be created by nodes, based on their direct interaction with other nodes

[†] This work was supported by the NS-CTA grant from the Army Research Laboratory. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARL.

or by using an intrusion detection system to monitor the behavior of nodes in their radio range [6]. Trust metadata created by individual nodes is required to be accessible to a network entity (e.g., the network administrator) for prompt decision making (e.g., revoking malicious nodes).

Trust metadata created by nodes could be propagated to a network entity that has a fixed location. However, since nodes geographically close to the entity could be involved in routing all packets containing trust metadata, their battery power could be used up excessively, that could lead to them being unable to participate in other network tasks. Also, since the entity is a centralized one, it would be unavailable to store trust metadata, if it crashes or is compromised by an adversary. Alternatively, nodes could create and store trust metadata in their own buffers. However, since a misbehaving node is aware of the identifiers of the nodes that come in its vicinity (the nodes that could have created trust metadata for its misbehavior), it could attempt to attack those nodes, resulting in either the loss of trust metadata stored in their buffers or trust metadata not being accessible to other nodes.

Because of the aforementioned concerns, in this work, we propose to use in-network storage for storing trust metadata. Based on its semantics, trust metadata created by a node will be propagated to and stored at a certain geographic location within the network itself. The propagation to the storage location could be done using a geographic routing algorithm (e.g., GPSR [7] routes the trust metadata to the node closest to the storage location, namely the storage node). If the storage node changes due to mobility, all trust metadata in the buffer of the old storage node could be transferred to the new storage node. Trust metadata would then always be available at its storage location in the network. A network entity such as a mobile trust authority (TA) could be assigned to perform trust aggregation [8], to evaluate the behavior of the nodes in the network. For this purpose, the TA could be online at certain times and issue queries in the network requesting for trust metadata. The requested trust metadata could be sent to the TA by the storage nodes.

However, there are security issues that interfere with the normal functioning of such a trust metadata management scheme. An adversary controlling the behavior of certain nodes (after node compromise), could attempt to prevent the trust metadata created for their misbehavior from being accessible to the TA. The compromised nodes could make trust metadata unavailable at a storage location (if the storage location of trust metadata is known to them). The compromised nodes could be present in the vicinity of a node to discover the storage location of the trust metadata that node propagates in the network. Packets containing trust metadata could also be dropped while routing, during propagation or retrieval of trust metadata, to or from a storage location, respectively.

In this paper, for security and scalability reasons, we propose a secure trust metadata management scheme to partition and store an information network of trust metadata of nodes in a MANET. That is, trust metadata is securely propagated to and stored at certain geographic locations inside the network, based on its semantics. The TA could then send different queries in the network to obtain the trust metadata of its interest. This scheme uses attribute-based encryption (ABE) [9, 10] for encryption/decryption of trust metadata, and is robust to several security attacks that attempt to disrupt the availability of trust metadata in the network. Our analysis shows that the proposed scheme provides desirable security and functionality properties with low query overhead.

The remainder of this paper is structured as follows. In Section 2, we present the system model and the background for trust metadata management in a MANET. In Section 3, we present two preliminary schemes and the proposed trust metadata management scheme. In Section 4, we analyze the simulation results of evaluation of the performance and robustness of the proposed scheme in comparison to the preliminary schemes. We then discuss the related work in Section 5 and finally, in Section 6, we state our conclusion and discuss future work.

2 System Model and Background

2.1 Network Model

We assume that the MANET area is divided into a set of equal sized regions in two dimensional space. For *each* region, the geographic location at its center is the storage location for trust metadata. Trust metadata is mapped to a storage location using a hash function that takes a location mapping key as input.

– *Trust Authority* : We assume the existence of a mobile trust authority (TA) in the MANET. The TA could be online at certain times and issue queries in the network to obtain the trust metadata of its interest, to evaluate the behavior of the nodes in the network.
– *Routing* : GPSR [7], a geographic routing algorithm, is used to propagate and retrieve trust metadata. A node involved in GPSR routing is required to be aware of its location (e.g., using a GPS) and the locations of the nodes in its radio range (a node sends periodic beacon messages with its location to its 1-hop neighbors). Additionally, a source node includes the destination location in any packet it sends.

2.2 Trust Metadata Management

We now introduce the concept of an in-network trust metadata management scheme. A node in a MANET could monitor the behavior of other nodes in its vicinity, e.g., using an intrusion detection system (IDS) [6], and map its observations into corresponding trust metadata. Trust metadata created by a node could include several components, namely, the *category* of misbehavior (e.g., jamming, packet dropping, etc.), the *observation region* (i.e., the region of the MANET in which misbehavior was observed), the *interval* (i.e., the time interval in which misbehavior was observed), the *evaluated node identifier* (i.e., the identifier of the misbehaving node), the *creating node identifier* (i.e., the identifier of the node that created the trust metadata), the *trust score* (e.g., a value between -1 and 1, where -1 denotes the least level of trust, 0 is neutral and 1 denotes the highest level of trust) [11, 12], the *evidence* (i.e., data to support the trust score, that could include the identifiers of a set of other witness nodes) and a *digital signature* (computed by the creating node over all the other components for authenticity and integrity). The category, observation region, interval, evaluated node identifier and creating node identifier are the components of trust metadata based on which the TA constructs its queries. We shall refer to these as the *query components* henceforth in this paper. We shall refer to the other components, namely the trust score, evidence and digital signature, as the *data components*.

A node could periodically create trust metadata for other nodes and propagate them in a message (an *update*) to a corresponding storage location in the network, determined based on the semantics of the trust metadata. Time could be divided into a set of intervals and trust metadata could be created and propagated at the beginning of each interval based on the observations during the previous interval. Trust metadata propagated to a storage location could be stored at the node closest to the storage location (the *storage node*). Trust metadata could also be propagated to multiple storage locations, if it is considered to be important (e.g., trust metadata for a node that is causing a serious denial-of-service attack), to improve its availability.

A storage node at a storage location could continuously monitor the locations of the nodes in its radio range. As soon as it detects that it no longer is the node closest to the storage location, the node closest to the storage location could become the new storage node. All trust metadata in the buffer of the storage node could be transferred to the new storage node in a set of messages. Thus, trust metadata would always be available at its storage location in the network. The TA could send a message (a *query*) to a storage location requesting for trust metadata. The storage node at the storage location could then send the requested trust metadata that it stores in its buffer in a set of messages (*replies*) to the TA. Different queries (see Section 2.3) could be issued by the TA, and the query communication overhead is required to be minimized.

2.3 Representative Queries

We now list some common representative queries that the TA could issue to obtain trust metadata :

- *Q1* : All trust metadata for a particular category of misbehavior, e.g., the list of nodes that performed jamming.
- *Q2* : All trust metadata for a particular category of misbehavior on a particular day (a set of intervals), e.g., the list of nodes that performed jamming in intervals 0 and 1.
- *Q3* : All trust metadata for a particular category of misbehavior on a particular day and in some areas of the network, e.g., the list of nodes that performed jamming in intervals 0 and 1 and in regions 0 and 1.
- *Q4* : All trust metadata for a particular node, e.g., all trust metadata created for the misbehavior of node N_2.

– *Q5* : All trust metadata by a particular node, e.g., all trust metadata created by node N_0.

2.4 Security Model

We assume that all nodes compromised by an adversary collude with one another and are able to eavesdrop only on the packets in their radio range. We assume that the TA has access to all the location mapping keys and is a trusted entity. The legitimates nodes could create and propagate trust metadata in the network for the misbehavior of the compromised nodes in their radio ranges. We term such legitimate nodes *observation nodes*. We assume that the goal of the adversary is to prevent the trust metadata created by the observation nodes for the misbehavior of the compromised nodes from eventually reaching the TA. In general, the compromised nodes could launch the following types of attacks :

– *Location control attack* : If the storage locations of the trust metadata created by an observation node are known to the adversary, it might attempt to make the trust metadata unavailable at those locations. At the time of propagation of trust metadata, the adversary attempts to ensure that the storage nodes at those storage locations are compromised nodes (e.g., a different compromised node stays close to each of the storage locations). The storage nodes then drop all the updates they receive.
– *Tailgating attack* : The compromised nodes are present in the radio range of an observation node at the time of propagation of trust metadata, eavesdrop on the packets forwarded by the observation node, attempt to identify the storage locations of the trust metadata in those packets and then launch location control attacks at those locations.
– *Selective dropping attack* : The compromised nodes selectively drop packets with trust metadata while routing, during propagation or retrieval of trust metadata. Also, the compromised nodes selectively drop trust metadata from their buffers when they become storage nodes.
– *Random dropping attack* : The compromised nodes randomly drop packets with trust metadata and trust metadata from their buffers with a certain probability.

2.5 Design Goals

We now present the design goals for a trust metadata management scheme in a MANET. The TA should be able to verify that the node that claims to have created some trust metadata actually created it (*authenticity*) and that the trust metadata is not modified subsequently (*integrity*). Only the TA and the creating node should be able to access trust metadata (*confidentiality*) and the identifier of the creating node should be known only to the TA (*source anonymity*). The TA should be able to receive replies to a query for trust metadata (*availability*). Different types of queries for trust metadata should be supported (*functionality*) and the query communication overhead should be minimized (*efficiency*).

3 Proposed Schemes

In this section, we first introduce two preliminary schemes and identify their drawbacks. We then present an overview, the details and security analysis of our secure trust metadata management scheme.

3.1 Preliminary Schemes

Scheme I : In this scheme, all trust metadata for a particular category of misbehavior is stored at the same storage location. The input to the hash function $H()$ that maps trust metadata for a category c to a storage location, is a location mapping key K_c, known to all the nodes in the network. Queries are issued to storage locations based on the categories of the requested trust metadata. Thus, propagation and retrieval of trust metadata is straightforward. Among the representative queries (Section 2.3), (Q1-Q3) are only sent to a particular storage location (they are requests for only a particular category of trust metadata) and (Q4-Q5) are sent to all the storage locations.

This scheme has several drawbacks. If a storage node crashes, all trust metadata for a particular category is lost. The location mapping keys are also known to an adversary. The adversary is also aware of the categories of trust metadata that could be created, since they are based on the misbehavior of the compromised nodes in its control. Thus, the storage locations of the trust metadata created by the observation nodes are known to the adversary and location control attacks could be launched at those locations. Tailgating attacks are effective by noting the storage location of trust metadata in headers of packets during propagation. Selective dropping attacks are effective since trust metadata is in clear text format in packets. Random dropping attacks are also effective, although multiple storage locations for a category of trust metadata could be used for robustness.

Scheme II : In this scheme, all trust metadata created by a particular node is stored at the same storage location. The input to the hash function $H()$ that maps trust metadata created by a node N_i to a storage location, is a location mapping key K_{N_i}, known only to node N_i and the TA. Thus, location control attacks due to knowledge of the location mapping keys are not possible. Trust metadata is encrypted for confidentiality. The query and the data components of trust metadata are separately encrypted by the creating node with its symmetric key shared with the TA. To send a query to a storage location, the TA creates the set of all tuples of the query components of the trust metadata it is requesting. It then encrypts each tuple in the set multiple times, each time with the symmetric key it shares with a node whose trust metadata is stored at the storage location, for all such nodes. When a storage node receives a query, it searches in its buffer for the encrypted tuples in the query, and returns all trust metadata for which there are matches to the TA.

Among the representative queries (Section 2.3), (Q1-Q4) are sent to all the storage locations, since the TA does not know which nodes have created trust metadata. Q5 is sent only to a particular storage location (it is a request for trust metadata created only by a particular node). The query overhead of this scheme is prohibitively high (as shown in Section 4.3), since the size of a query is very big and a query is sent to all the storage locations (except Q5).

3.2 Overview

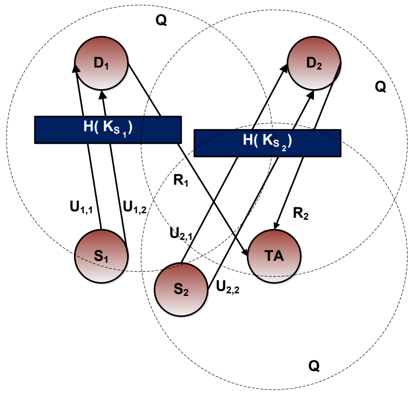


Fig. 1. Proposed Scheme Overview : $U_{1,1}$ and $U_{1,2}$ are updates created by node S_1 and $U_{2,1}$ and $U_{2,2}$ are updates created by node S_2. Q is a query by the TA that is broadcast in the network and R_1 and R_2 are the sets of replies to Q from storage nodes D_1 and D_2 respectively.

We now present an overview of the proposed scheme. To achieve confidentiality, trust metadata is encrypted by the creating nodes. A storage node is required to search in its buffer for the trust metadata requested in a query and send the trust metadata for which there are matches to the TA. However, a storage node is unable to access the encrypted trust metadata in its buffer. A storage node could send all trust metadata it stores to the TA, irrespective of which query it receives, a clearly inefficient solution. Including all encrypted tuples of the query components of requested trust metadata in a query (preliminary scheme II) is also inefficient, due to very high query size. To meet the seemingly contradictory requirements of being able to process a query without access to trust metadata simultaneously, we propose the notion of *decryption on-demand* for query processing. This requires support for a different type of encryption, i.e., attribute-based encryption (ABE) [9, 10]. In ABE, encrypted data has some descriptive attributes and a decryption key is associated with an access structure. Decryption is possible only if the attributes of the encrypted data match the access structure of the key. The values of the query components of trust metadata could be chosen as the attributes.

Specifically, a creating node first encrypts trust metadata with its symmetric key shared with the TA and then encrypts further with ABE. A node periodically encloses such encrypted trust metadata it creates for other nodes in an update and propagates it to its storage location. The digital signature component of trust metadata provides authenticity and integrity. Symmetric encryption of trust metadata provides confidentiality. The source node identifier is not included in the headers of packets with trust metadata during propagation. Due to this and symmetric encryption of trust metadata, source anonymity is achieved and selective dropping attacks are prevented. The header of a packet with trust metadata is encrypted hop-by-hop during propagation, to hide the storage location and thus prevent tailgating attacks. Each node uses a private location mapping key (shared with the TA), thereby preventing location control attacks. Additionally, techniques such as hop-by-hop payload shuffle [13] could be used to prevent matching of contents of packets to determine the mapping of trust metadata created by nodes to their storage locations. Availability is affected only by random dropping attacks. However, we show in Section 4.3 that this scheme is robust even to random dropping attacks, if redundant storage locations are used for updates.

The TA broadcasts a query in the network with an ABE decryption key (along with a digital signature that is verified by the storage node), that would only decrypt the trust metadata it is requesting. If a storage node is able to decrypt a set of trust metadata in its buffer with the key (it is possible to determine if decryption is successful or not, as we note in Section 3.3), it sends them in replies to the TA (each trust metadata in the set is still encrypted with symmetric encryption). Note that the storage node would not know what trust metadata was requested in the query, even if it is able to process it. The TA is then able to decrypt (with its symmetric key shared with the creating node) and verify the digital signature of all trust metadata in the replies. Note that with just a single broadcast query, all the storage nodes are able to check if they store the trust metadata requested in the query and send them to the TA if they do. The TA is able to create specific decryption keys for the trust metadata it is requesting (see Section 3.3) and thus, functionality is achieved. Among the representative queries (see Section 2.3), (Q1-Q4) are broadcast in the network and Q5 is sent only to a particular storage location (it is a request for trust metadata created only by a particular node). The query overhead of this scheme is much lower and thus the efficiency is much higher compared to preliminary scheme II, since the average size of a query is much lower. Figure 1 illustrates the overall working of the proposed scheme.

3.3 Details of Proposed Scheme

We now discuss the details of encryption and decryption using ABE in the proposed scheme. We use the construction for access-trees for ABE proposed in [9]. In the access-tree construction, encrypted data is labeled with some attributes. A decryption key is identified by a tree-access structure in which each interior node of the tree is a threshold gate (a t-of-n threshold gate returns TRUE if and only if at least t of the n inputs are TRUE, OR is a 1-of-n gate and AND is an n-of-n gate) and the leaves are associated with attributes. Decryption is possible if and only if there is an assignment of the attributes from the encrypted data to the nodes of the tree such that the tree is satisfied. For details of the access-tree construction for ABE, please refer to [9].

Preliminaries We define the following functions related to access-trees. $num(x)$ of a node x in the tree is its number of children. The children of a node x are numbered from 1 to $num(x)$ in an arbitrary manner. $index(x)$ of a node x is such a number associated with node x (denoting its number for its parent node). $parent(x)$ of a node x is its parent node. $att(x)$ of a leaf node x is the attribute associated with node x. $k(x)$, a threshold value, of a leaf node x is 1, and of a non-leaf node x is t, if the node is a t-of-$num(x)$ threshold gate.

We now present some facts about groups with efficiently computable bilinear maps. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_1$ and e be a bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degeneracy: $e(g, g) \neq 1$

We say that $\mathbb{G}_1$ is a bilinear group if the group operation in $\mathbb{G}_1$ and the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ are both efficiently computable.

Algorithm 1 Creation of the Public Parameters and the Master Key by the TA

```
1: procedure setup($\mathcal{U}$)
2: for each attribute $i \in \mathcal{U}$ do
3:     choose a number $t_i$ uniformly at random from $\mathbb{Z}_p$
4: end for
5: choose a number $y$ uniformly at random from $\mathbb{Z}_p$
6: the public parameters PK are
       $T_1 = g^{t_1}, \cdots, T_{|\mathcal{U}|} = g^{t_{|\mathcal{U}|}}, Y = e(g, g)^y$
7: the master key MK is
       $t_1, \cdots, t_{|\mathcal{U}|}, y$
8: end procedure
```

Setup Let $\mathbb{G}_1$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. Define the universe of attributes of trust metadata $\mathcal{U} = \{1, 2, \cdots, n\}$. The TA creates the public parameters and the master key for ABE at setup. Algorithm 1 lists the steps.

Trust Metadata Creation Algorithm 2 lists the steps for trust metadata creation and encryption. A node N_i creates trust metadata TM and encrypts it with its symmetric key shared with the TA to obtain TM'=$E(TM, K_{N_i,TA})$. Node N_i then encrypts TM' ($TM' \in \mathbb{G}_2$) with ABE, based on the set of attributes γ. The values of the query components of TM are chosen as the attributes that belong to γ. The number of exponentiations for encryption is about the same as the number of attributes in γ [9].

Algorithm 2 Encryption of Trust Metadata by the Creating Node

```
1: procedure $encryption(TM, \gamma, PK, K_{N_i,TA})$
2:
       $$TM' = E(TM, K_{N_i,TA})$$
3: choose a number $s$ uniformly at random from $\mathbb{Z}_p$
4: the encrypted trust metadata is
       $$ETM = (\gamma, E' = TM'Y^s, \{E_i = T_i^s\}_{i \in \gamma}).$$
5: end procedure
```

Algorithm 3 Creation of a Decryption Key by the TA

```
1: procedure $decryption\_key(\mathcal{T}, MK, x)$
2: degree $d_x$ of a polynomial $q_x$ for node $x \leftarrow k(x) - 1$
3: if $x$ is the root node of $\mathcal{T}$ then
4:     $q_x(0) \leftarrow y$
5: else
6:     $q_x(0) = q_{parent(x)}(\text{index}(x))$
7: end if
8: choose $d_x$ other points of $q_x$ at random to define it completely
9: if $x$ is a leaf node of $\mathcal{T}$ then
10:
       $$D_x \leftarrow g^{\frac{q_x(0)}{t_{att(x)}}}$$
11: end if
12: for each node $z \in \mathcal{T}$ such that $parent(z) = x$ do
13:     call $decryption\_key(\mathcal{T}, MK, z)$
14: end for
15: end procedure
```

Query Creation The TA creates a decryption key based on the access-tree for the trust metadata it is requesting by applying the recursive procedure shown in Algorithm 3 with the access-tree, its root node and the master key as inputs. The procedure calculates a secret value D_x for each leaf node x in the tree. The set of such secret values is the decryption key D. Figure 2 shows examples of the access-trees for the representative queries defined in Section 2.3.

Query Processing To process a query by the TA, a storage node applies the recursive procedure shown in Algorithm 4 with an encrypted trust metadata in its buffer, the decryption key in the query and the root of the access-tree as inputs (we assume that the access-tree $\mathcal{T}$ is embedded in the decryption key). The procedure returns a group element of $\mathbb{G}_2$, $e(g,g)^{ys} = Y^s$, if decryption is successful. In this case, TM', the trust metadata encrypted with the symmetric key shared by the creating node and the TA, is obtained by dividing E' (a part of ETM) by Y^s, and is sent by the storage node to the TA. The procedure returns $\perp$ if decryption is unsuccessful and in this case, nothing is sent by the storage node. The number of pairing computations and exponentiations for decryption

Algorithm 4 Processing of a Query by a Storage Node

```
1: procedure $decryption(ETM, D, x)$
2: if $x$ is a leaf node of $\mathcal{T}$ then
3:     if $att(x) \in \gamma$ then
4:         return $e(g, g)^{s \cdot q_x(0)}$
5:     else
6:         return $\perp$
7:     end if
8: else
9:     $count \leftarrow 0$
10:    for each node $z \in \mathcal{T}$ such that $parent(z) = x$ do
11:        $res = decryption(ETM, D, z)$
12:        if $res \neq \perp$ then
13:            $count \leftarrow count + 1$
14:        end if
15:    end for
16:    if $count \geq k(x)$ then
17:        return $e(g, g)^{s \cdot q_x(0)}$
18:    else
19:        return $\perp$
20:    end if
21: end if
22: end procedure
```

could be reduced from the number of nodes in the access-tree to a minimal set of its leaf nodes [9].

Example of ABE Encryption/Decryption Let us consider an example of ABE encryption/decryption. Let $\mathcal{U}$ be $\{1, 2, \cdots, 18\}$. Let the query components of trust metadata TM_0 created by a node N_0 be $(jamming, 0, 0, N_2, N_0)$ and the corresponding set of attributes γ_0 be $\{1, 3, 7, 13, 15\}$. Let the query components of TM_1 by N_1 be $(jamming, 1, 2, N_2, N_1)$ and γ_1 be $\{1, 4, 9, 13, 16\}$. Consider the access-tree for query Q3 in Figure 2. In general, an internal node x in the access-tree that is a $k(x)$-of-$num(x)$ gate does not return $\perp$ if at least $k(x)$ of its children do not return $\perp$. For TM_0, the *Decryption* function returns $\perp$ for the node Interval:1 and the node Observation Region:1 but does not return $\perp$ for any of the other nodes including the root node and thus, decryption is possible with the decryption key associated with the access-tree. However for TM_1, the *Decryption* function returns $\perp$ for the node Interval:0, the node Interval:1, their parent node (the OR gate), the node Observation Region:0 and the root node and thus, decryption is not possible with the decryption key associated with the access-tree. Recall that each trust metadata is still encrypted with symmetric encryption even after ABE decryption and a storage node is unable to access trust metadata. Also, a storage node does not know what trust metadata is requested in a query, even if it is able to process it.

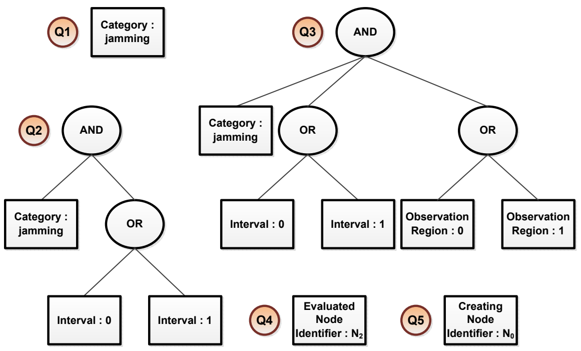


Fig. 2. Examples of Access-Trees for Representative Queries (Q1-Q5)

Table 1. Security Analysis

Security Property/Attack	*Comments*
Authenticity and Integrity	Digital signature component of trust metadata
Confidentiality	Symmetric encryption of trust metadata
Source Anonymity	Symmetric encryption of trust metadata and not including the source node identifier in headers of packets with trust metadata during propagation
Availability	Security property satisfied by the same way robustness to all the attacks listed below is achieved
Location control attacks	Private location mapping keys for nodes
Tailgating attacks	Hop-by-hop encryption of headers of packets with trust metadata during propagation
Selective dropping attacks	Robustness achieved by the same way source anonymity is satisfied
Random dropping attacks	Redundant storage locations for updates

3.4 Security Analysis

We summarize how the security properties are satisfied and robustness to attacks is achieved by the proposed scheme in Table 1.

4 Performance Evaluation

In this section, we evaluate the performance and robustness to *random dropping attacks* of the proposed trust metadata management scheme in comparison to the two preliminary schemes. Note that none of the other possible attacks discussed in Section 2.4 are effective in our proposed scheme, whereas they are effective in the two preliminary schemes. However, we evaluate only the effect of random dropping attacks on these schemes for comparison.

4.1 Experiment Setup

We perform our evaluation using the GloMoSim [14] simulator. The geographic routing algorithm we use for our evaluation is GPSR [7], that we have implemented in GloMoSim. We have also implemented a trust management framework at the application layer. The network area is divided into 9 regions and the node closest to the center of each region (that changes based on mobility) is assigned to store trust metadata. We simulate 50 nodes (each with a 802.11 radio range of 100m) in a 300x300m network following the random waypoint mobility model with minimum speed 1m/s, maximum speed 20m/s and pause time 60s.

The TA sends a query in the network, at a random time during each interval, to obtain trust metadata. We assume that a query by the TA is either Q1 or Q4 (Q2 and Q3 are subsets of Q1 and Q5 is not a broadcast query in the proposed scheme, the representative queries are defined in Section 2.3). The size of an ABE decryption key depends on the number of leaf nodes in the corresponding access-tree. For queries Q1 and Q4, there is just a single leaf node. We simulate 2 categories of trust metadata. We randomly assign C percent of all nodes to be compromised nodes that participate in random dropping attacks, with a probability of random dropping p. Each point in the figures is the mean of the results obtained using 25 different randomly generated motion patterns with different seed values.

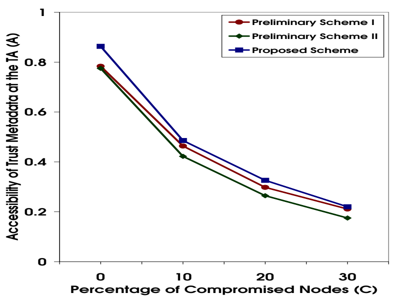
4.2 Metrics

We define the following metrics to evaluate a trust metadata management scheme:

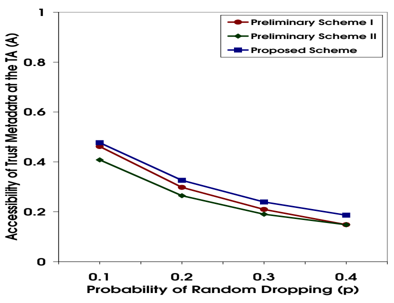
- *Accessibility of trust metadata at the TA (A)* : The fraction of all trust metadata requested by the TA in a query that it obtains, averaged over all times when queries are issued. Ideally, the value of A should be close to 1.
- *Query communication overhead in bytes (B)*: The number of bytes transmitted in the network for trust metadata queries by the TA. The value of B should be minimized.

4.3 Simulation Results

Figure 3(a) shows the variation of the accessibility of trust metadata at the TA (A) with the percentage of compromised nodes (C). Here, the C compromised nodes perform random dropping attacks with a certain probability $p = 0.2$. As C increases, A decreases for all three schemes. This trend is expected since the more the number of compromised nodes launching random dropping attacks, the more the chances of trust metadata requested by the TA not reaching it. Note that A for the proposed scheme is higher compared to the preliminary schemes. This is so since a query is broadcast in the network in the proposed scheme and thus, the likelihood of a query reaching a storage location is high. On the other hand, in the preliminary schemes, if a unicast query is dropped or is lost in the network before it reaches a storage location, the TA would not obtain any replies to the query from the storage location.



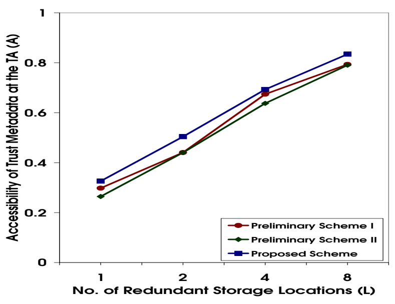
(a)



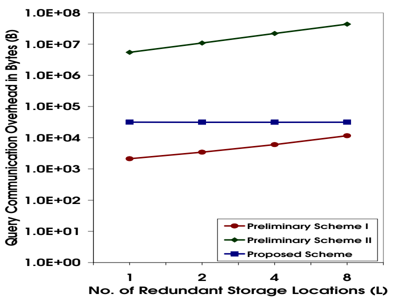
(b)

Fig. 3. (a) Variation of the Accessibility of Trust Metadata at the TA (A) with the Percentage of Compromised Nodes (C), the Probability of Random Dropping (p) = 0.2 (b) Variation of A with p, $C = 20\%$

Figure 3(b) shows the variation of A with the probability of random dropping (p). Here, $C = 20\%$. As p increases, A decreases for all three schemes. This trend is expected since the more the probability of random dropping, the more the chances of trust metadata requested by the TA not reaching it. Note that once again, A for the proposed scheme is higher compared to the preliminary schemes, due to the usage of broadcast queries.



(a)



(b)

Fig. 4. (a) Variation of A with the number of redundant storage locations (L), $C = 20\%$, $p = 0.2$ (b) Variation of the Query Communication Overhead in Bytes (B) with L, $C = 20\%$, $p = 0.2$ (the y-axis is in logarithmic scale)

For a trust management scheme to be robust to random dropping attacks, an update could be propagated to multiple (redundant) storage locations. Queries could be unicast to each of the multiple storage locations separately (preliminary schemes) or broadcast in the network (proposed scheme). Replies to queries could then be obtained from each of the multiple storage locations. Figure 4(a) shows the variation of A with the number of redundant storage locations (L). Here, $C = 20\%$ and $p = 0.2$. As L increases, A increases for all three schemes. This trend is expected since the more the number of redundant storage locations for an update, the more the chances of trust metadata requested by the TA reaching it.

The variation of the query communication overhead in bytes (B) with L is shown in Figure 4(b). The y-axis in this figure is in logarithmic scale. Note that we consider the query overhead and not the total overhead of trust metadata management as a metric. This is so since the message overhead for updates, replies and change in storage nodes is similar across the three schemes on an average, but is different for queries, based on how efficient they are. As expected, preliminary scheme II has a very high query overhead compared to the other two schemes (see Sections 3.1 and 3.2). The query overhead of the preliminary schemes increases with L. Even though B is lower for preliminary scheme I compared to the proposed scheme, preliminary scheme I has some serious limitations and is not secure (as noted in Section 3.1). Note that for the proposed scheme, B remains about the same as L increases, due to the usage of broadcast queries, irrespective of the value of L. Thus, the proposed scheme is more scalable compared to the preliminary schemes.

5 Related Work

To the best of our knowledge, this paper proposes the first scheme for in-network storage of trust metadata in a mobile ad-hoc network, that also satisfies a number of security properties and is robust to several security attacks that attempt to disrupt the availability of trust metadata in the network.

A trust management framework could be subjected to other types of security attacks [2, 15, 12]. For example, bad mouthing attack, in which dishonest recommendations are provided by nodes, on-off attack, in which nodes behave well and badly alternatively, conflicting behavior attack, in which nodes behave differently with different peer nodes, sybil attack, in which several fake identifiers are created by a node and newcomer attack, in which a node registers itself as a new user. Li et al. [2] and Sun et al. [15] identified such attacks and proposed defense techniques. Our current work could be used in conjunction with the defense techniques proposed in [2, 15] to improve the overall security of a trust management scheme in an MANET.

Li et al. [1] described a multi-dimensional trust evaluation framework from different perspectives, namely collaboration trust for selfish behavior, behavioral trust for malicious behavior and reference trust for opinion correctness. They proposed performing different types of independent observations to obtain the trust metadata for each dimension. Balakrishnan et al. [5] proposed coupling a trust management framework with other security models such as key management and secure routing. They recommend security models provide feedback to one another to improve the overall security of the network.

In some previous protocols [4, 3], nodes themselves initiate requests to obtain trust metadata created by other nodes (recommendations). Requests are sent and replies are received based on the recommendation trust values of nodes, i.e., trust metadata for providing correct recommendations. A node evaluates recommendation trust by comparing recommendations to its own observations. In [16], a trust metadata storage system was proposed. However, this system does not consider security attacks at all. Thus, it is vulnerable to the attacks mentioned in this paper.

6 Conclusion and Future Work

In this paper, we propose an in-network secure trust metadata management scheme for a mobile ad-hoc network. We identify the security and performance design goals for such a scheme and note that the proposed scheme satisfies those goals. We also evaluate the performance and robustness of the proposed scheme and observe that it compares favorably to two preliminary schemes.

In the future, we shall devise techniques to obtain the location proof of a node that creates trust metadata in a mobile ad-hoc network. The purpose is to detect false claims by nodes of their presence in particular regions of the network and observation of misbehavior of nodes to create trust metadata.

References

1. Li, W., Joshi, A., Finin, T.: Coping with Node Misbehaviors in Ad Hoc Networks : A Multi-Dimensional Trust Management Approach. In: Proc. IEEE MDM. (2010)
2. Li, J., Li, R., Kato, J.: Future Trust Management Framework for Mobile Ad Hoc Networks. In: Proc. IEEE Communications Magazine. Volume 46. (2008) 108–114
3. Sun, Y., Yu, W., Han, Z., Liu, K.: Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks. In: Proc. IEEE Journal on Selected Areas in Communications. Volume 24. (2006) 305–317
4. Velloso, P., Laufer, R., Cunha, D., Duarte, O., Pujolle, G.: Trust Management in Mobile Ad Hoc Networks Using a Scalable Maturity-Based Model. In: Proc. IEEE Transactions on Network and Service Management. Volume 7. (2010) 172–185
5. Balakrishnan, V., Varadharajan, V., Tupakula, U., Lues, P.: TEAM : Trust Enhanced Security Architecture for Mobile Ad-Hoc Networks. In: Proc. IEEE ICON. (2007)
6. Zhang, Y., Lee, W.: Intrusion Detection in Wireless Ad-Hoc Networks. In: Proc. ACM MobiCom. (2000)
7. Karp, B., Kung, H.: GPSR : Greedy Perimeter Stateless Routing for Wireless Networks. In: Proc. ACM MobiCom. (2000)
8. Wang, Y., Singh, M.: Trust Representation and Aggregation in a Distributed Agent System. In: Proc. ACM AAAI. (2006)
9. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In: Proc. ACM CCS. (2006)
10. Yu, S., Ren, K., Lou, W.: FDAC: Toward Fine-Grained Distributed Data Access Control in Wireless Sensor Networks. In: Proc. IEEE INFOCOM. (2009)
11. Falcone, R., Pezzulo, G., Castelfranchi, C.: A Fuzzy Approach to a Belief-Based Trust Computation. In: Proc. ACM AAMAS. (2002)
12. Govindan, K., Mohapatra, P.: Trust Computations and Trust Dynamics in Mobile Adhoc Networks : a Survey. In: Proc. IEEE Communications Surveys and Tutorials. Volume 14. (2011) 279–298
13. Zhu, B., Wan, Z., Kankanhalli, M., Bao, F., Deng, R.: Anonymous Secure Routing in Mobile Ad-Hoc Networks. In: Proc. IEEE Local Computer Networks. (2004)
14. GloMoSim: Global Mobile Information Systems Simulation Library. http://pcl.cs.ucla.edu/projects/glomosim
15. Sun, Y., Han, Z., Liu, K.: Defense of Trust Management Vulnerabilities in Distributed Networks. In: Proc. IEEE Communications Magazine. Volume 46. (2008) 112–119
16. Natarajan, V., Zhu, S., Srivatsa, M., Opper, J.: Semantics-Aware Storage and Replication of Trust Metadata in Mobile Ad-Hoc Networks. In: Proc. IEEE AINA. (2012)