# SMART: An SVM-based Misbehavior Detection and Trust Management Framework for Mobile Ad hoc Networks

Wenjia Li, Anupam Joshi, and Tim Finin

**Abstract**—Due to lack of pre-deployed infrastructure, nodes in Mobile Ad hoc Networks (MANETs) are required to relay data packets for other nodes to enable *multi-hop* communication between nodes that are not in radio range with each other. However, whether for self ish or malicious purposes, a node may refuse to cooperate during the network operations or even attempt to interrupt them, both of which are recognized as misbehaviors. In this paper, we describe a SVM-based Misbehavior Detection and Trust Management framework (*SMART*) to address the security threats caused by various misbehaviors. In SMART, the Support Vector Machine algorithm is used to detect node misbehaviors, which does not require any pre-def ined threshold to distinguish misbehaviors from normal behaviors. In addition, multi-dimensional trust management scheme is applied evaluate the trustworthiness of MANET node from multiple perspectives, allowing the trustworthiness of each node to be described in a more accurate and effective manner. To validate the SMART framework, an extensive performance study is conducted using simulation. The results show that the SMART framework outperforms previous schemes: It handles a larger fraction of adversaries, and it is resilient when nodes are highly mobile. More importantly, it can detect adversaries that alter their behaviors over time.

**Index Terms**—Security, Misbehavior Detection, Multi-dimensional Trust Management, Mobile Ad hoc Networks.

## 1 Introduction

A Mobile Ad-hoc NETwork (MANET) generally refers to a network without any pre-deployed network infrastructure in which the mobile nodes in the network dynamically set up the paths among themselves and relay packets for other nodes. MANETs have a variety of both civilian and military applications, ranging from supporting disaster relief personnel in coordinating rescue efforts after a hurricane, earthquake or brush fire [1] to soldiers exchanging information for situational awareness on the battlefield [2]. Other possible applications include mobile healthcare system [3], real-time traffic alert propagation via vehicular networks [4], and Cyber-Physical System (CPS) monitoring.

Security is a key concern in MANETs because their nodes are generally more susceptible to various threats than those in traditional wired networks. In this paper, an SVM-based Misbehavior Detection and Trust Management framework (*SMART*) is described to secure MANETs. In this framework, the Support Vector Machine algorithm is used to train a classifier, and then the classifier is used to detect misbehaviors. In this way, misbehaviors can be identified in a more adaptive manner when compared to the traditional threshold-based mechanisms [5], [6]. Furthermore, the trustworthiness of each node is modeled as a vector rather than a single scalar so that it can be evaluated in a more accurate and effective way. We show that the SMART framework outperforms the previous mechanisms via an extensive performance study.

• *The authors are with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC), Baltimore, MD, 21250.*
*E-mail: {wenjia1,joshi,f inin}@cs.umbc.edu*

## 2 Background and Motivation

Security systems in MANETs differ significantly from those in the traditional wired networks in four important ways.

- Open transmission medium: data in MANETs are transmitted via radio frequency (RF) broadcasts. The open nature of the RF signals makes eavesdropping easier.
- Absence of pre-deployed infrastructure: lacking a centralized network infrastructure, cooperation among the mobile nodes becomes an important precondition for the security systems in MANETs. These security systems in MANETs are required to be more resilient to uncooperative behaviors than in wired networks.
- Short transmission range: due to the limited battery power, the transmission range among the nodes is generally restricted, which makes it difficult for nodes to obtain a global view of what is happening in the whole network from its own observations. Hence, each node needs to rely on the observations from other nodes to fully understand what is happening in the network.
- High mobility of the nodes: being equipped with wireless transmission devices, mobile nodes are free

to make any arbitrary movement. This can occasionally interrupt communication because nodes may move out of the transmission range of each other from time to time during communication processes.

Due to the features listed above, MANETs are much more vulnerable to a variety of node misbehaviors when compared to the traditional wired networks. Unfortunately, the traditional security solutions for the wired networks are not directly applicable to MANETs, as have been well studied in literature [7], [8], [9], [10], [11], [12], [13].

Since node misbehaviors can do great harm to MANETs, numerous security solutions have been proposed to detect and mitigate those misbehaviors from a variety of perspectives. Among these security solutions, the misbehavior detection method is a well-known countermeasure to fight against node misbehaviors [14], [5], [8]. The majority of existing misbehavior detection methods rely on a well-defined *threshold* to identify the possible misbehaviors. However, due to high mobility of the nodes as well as short transmission range, the *normal* behavioral patterns of the well-behaved nodes can change rapidly. Therefore, it is not feasible to set an appropriate threshold for misbehaviors that works well all the time. Furthermore, a *smart* adversary can adjust its behavior from time to time, i.e., it occasionally alters the type and quantity of the misbehaviors that it conducts to remain below the detection threshold [9], [10].

Trust management schemes are widely-used and closely coupled with the misbehavior detection schemes. Because it is quite useful to evaluate a node's behaviors and determine if it is trustworthy in terms of how cooperative it is, trust management schemes have become a powerful tool to deal with node misbehaviors. A variety of trust management mechanisms have been studied during the past decade, such as the mechanisms discussed in [7], [15], and [16]. Most of them model the trust of a node in one dimension, i.e., all available evidences and observations are used to calculate a single scalar trust metric for each node. However, a single scalar trust measure may not be expressive enough to accurately characterize whether a node is trustworthy or not in many complicated scenarios. Figure 1 shows an example scenario in which a single scalar trust is not expressive enough.

From Figure 1, we see that in the first step (a), the observer collects and records the misbehaviors that are conducted by nodes 1, 2, and 3. The observation results illustrate that the nodes have modified packets, spread incorrect opinions regarding others (for example, intentionally falsely accusing other nodes of dropping packets) and sent continuous Request-To-Send (RTS) frames at a same amount of 10, respectively. Suppose that these three types of misbehaviors are punished at the same rate when the trustworthiness of each node is evaluated. Then, in the next step (b), the observer may draw a conclusion that all these three nodes are equally untrustworthy. As a result, the observer will treat

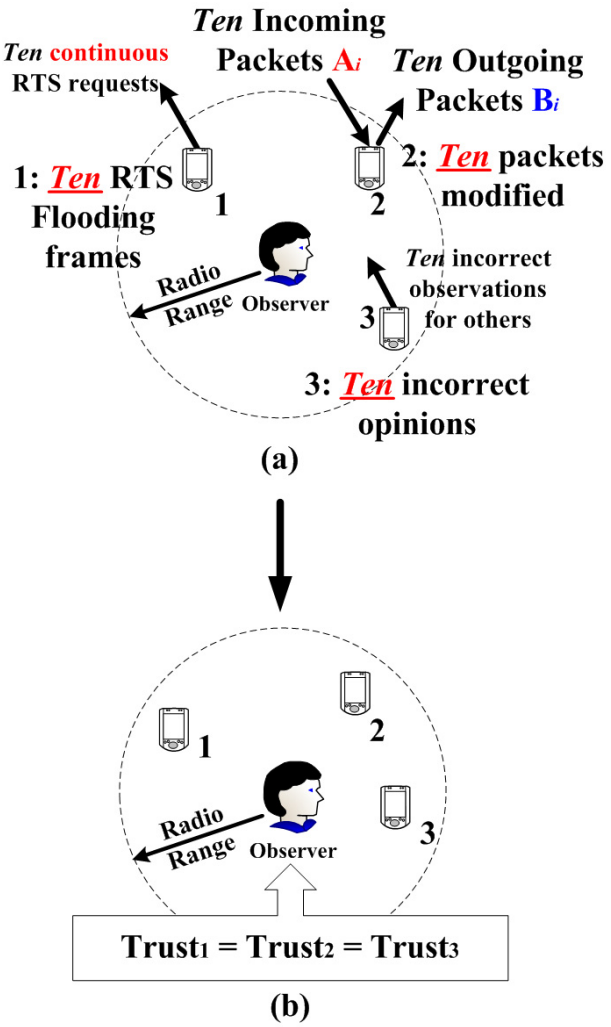



Fig. 1. An example scenario where a single scalar trust is NOT expressive enough.

node 2 and node 3 equally when it needs to determine which node to forward packets as well as which node it should believe when exchanging opinions. However, it is obvious that the trustworthiness of node 2 and node 3 is not equivalent when it comes to both packet forwarding and opinion exchanging. We argue that it is neither accurate nor effective to merely use a single scalar when the trustworthiness of a node is evaluated.

In this paper, a SVM-based Misbehavior Detection and Trust Management framework (a.k.a *SMART*) is proposed to better address the security vulnerabilities that are caused by various node misbehaviors. Our major contributions are as follows:

- **A novel SVM-based misbehavior detection method.** In SMART, the Support Vector Machine (SVM) method [17] is used in the misbehavior detection scheme to build a classifier to distinguish misbehaving nodes from well-behaved nodes. The misbehavior detection method used in SMART does not require any pre-determined threshold to determine

misbehaving nodes. Therefore, the misbehavior detection method works well in complicated scenarios where network topology and/or adversary model can change remarkably.

- **Evaluating the trustworthiness of mobile nodes from several well-defined perspectives.** The multi-dimensional trust management scheme is deployed in SMART, in which the notion of trustworthiness is further classified into several attributes (i.e., dimensions). Each attribute is able to indicate whether or not a node is trustworthy in terms of one specific feature that it should be, such as cooperative, well-behaved, and honest. The initial version of this scheme has been discussed in our previous work [12].
- **An *adaptive* trust evolution model in which each dimension of trustworthiness is adjusted at a different pace.** Each dimension of the trustworthiness can be adjusted according to the features of the misbehavior(s) to which the dimension is related, such as severity of the outcome, frequency of occurrence, and context in which the misbehavior(s) occur.

The rest of the paper is organized as follows. In Section 2, we briefly review the literature on misbehavior detection as well as trust management for MANETs. Next, we present the design goals and the system model in Section 3, followed by a detailed discussion to the proposed SMART framework in Section 4. Then, a comprehensive simulation study is given in Section 5 to evaluate the performance of the SMART framework, and the paper is finally concluded in Section 6.

## 3 Related Work

There is rich literature on the topics of misbehavior detection as well as trust management for ad hoc networks. Hence, the related work for these two research topics will be discussed separately in this section.

### 3.1 Misbehavior Detection for Ad hoc Networks

When it comes to the discussion of misbehavior detection, we should first clearly understand the term *misbehavior* itself. The term *misbehavior* generally refers to a group of abnormal behaviors that deviates from the set of behaviors that each node is supposed to conduct in MANETs [18]. In general, misbehaviors can occur at each layer in MANETs, such as (1) malicious flooding of the RTS frames in the MAC layer, (2) drop, modification, and misroute to the packets in the network layer, and (3) deliberate propagation of fake opinions regarding the behaviors of other nodes in the application layer.

Node misbehaviors may range from lack of cooperation to active attacks aiming to cause Denial-of-Service (DoS) and interruption of the network operations. According to [19], there are four types of misbehaviors in ad hoc networks, namely *failed node behaviors*, *badly failed node behaviors*, *selfish attacks*, and *malicious attacks*. These four types of misbehaviors are classified with respect to the node's intent and action. Selfish attacks are intentional passive misbehaviors, where nodes choose not to fully participate in the packet forwarding functionality to conserve their resources, such as battery power. Malicious attacks are intentional active misbehaviors, where the malicious node aims to purposely interrupt network operations. For instance, because of the limited battery power that each node possesses, a *selfish* node may choose not to cooperate with other nodes so as to preserve its own battery power [20]. In other words, when a *selfish* node is asked to forward some data packets for other nodes, it may choose to drop either a part or all of the incoming packets. By this means, it can save its battery power and thus transmit some extra packets for the sake of itself. In contrast, the *malicious* nodes aim to intentionally disturb the network services, and they may deliberately drop, modify or misroute packets while their primary concern is not battery life [21]. Regardless of the reason for a node's misbehavior, it degrades the MANET's functioning.

An Intrusion Detection System (IDS) is normally regarded as an important solution for detecting various node misbehaviors in MANETs. Several approaches have been proposed to build IDS probes on each mobile node due to the lack of a fixed infrastructure [14], [22], [23]. In these approaches, there is one IDS probe installed on each node that monitors the network traffic. Then, each node may cooperate with other nodes to further refine the detection results from time to time. On the other hand, Huang et al. [24] proposed a cooperative IDS framework, in which clusters are formed and a node in each cluster acts as the cluster head in turn and coordinate among all the cluster members for the intrusion detection process. In addition, Parker et al. [6] proposed a cross-layer intrusion detection method in which evidences for misbehaviors are collected and then combined at multiple layers. By this means, the observations from multiple layers are integrated in case that they are related to the same misbehavior, and the misbehaviors can be *amplified* in presence of various ambient noises, such as abnormal behaviors caused by radio interference and congestion.

Routing misbehaviors are another major security threats that have been extensively studied in ad hoc networks. In addition to externally intruding into MANETs, an adversary may also choose to compromise some nodes in ad hoc networks, and make use of these internal resources to disturb the routing services so as to make part of or the entire network unreachable. Marti et al. [5] introduced two related techniques, namely *watchdog* and *pathrater*, to detect and isolate misbehaving nodes, which are nodes that do not forward packets for others. Other solutions aim to cope with various routing misbehaviors [25], [26], [27].

The major differences between our misbehavior detection method and related methods are as follows.

First, we adopt an abnormal behavior detection and

behavioral data preparation method using ***outlier detection*** technique. As in our previous work [9], [10], [11], [12], we apply the outlier detection technique in SMART to identify abnormal behaviors, which generally appear as outliers when compared to normal behaviors. Unlike our previous work, however, we use these abnormal behaviors as well as the normal behaviors to train a SVM classifier instead of directly viewing outliers as misbehaviors. In addition, the detection of abnormal behaviors neither relies on any pre-defined normal behavioral pattern, nor does it require a predefined threshold to distinguish normal behaviors from abnormal behaviors. Hence, the proposed misbehavior detection method can better adapt to complicated application scenarios such as instable network topology and constantly changing abnormal behavioral pattern of *smart* adversaries.

Second, our misbehavior detection method adaptively factors in the *trustworthiness* of the mobile nodes. Due to the short transmission range and high mobility of the nodes, it is neither practical nor energy-efficient for each node to keep track of the behaviors of all other nodes. Therefore, in order to completely detect all the misbehaviors in the network, each node needs to partially rely on the observations for misbehaviors that are made by others. However, incorrect or even falsified observations may be propagated among the nodes either due to out of ignorance or because of malicious intent. Therefore, it is essential for each node to decide how to appropriately integrate the observations from other nodes. On the other hand, most misbehavior detection methods have not taken the correctness of the observations into account when they are aggregated. For instance, Zhang et al. stated in [14] that the adversaries generally have no incentives to send incorrect reports of intrusions because they may be excluded from the network. However, as has been described by Buchegger et al. [28], the adversaries sometimes may choose to spread false observations to accuse other nodes so that their misbehaviors may remain undetected. Hence, we introduce the notion of *trustworthiness* to help assess the observations reported by other nodes during the misbehavior detection process.

### 3.2 Trust Management for MANETs

The main goal of trust management is to evaluate behaviors of other nodes and assign a reputation to each node based on the behavior assessment. The reputation can be used to decide the trustworthiness for other nodes, make choices on which nodes to cooperate with, and even take action to punish an untrustworthy node if needed.

In general, a trust management scheme relies on two types of observations to evaluate the node behaviors. The first kind consists of *first-hand* or *direct* observations [29]. *Second-hand* or *indirect* observations are generally obtained by exchanging both direct and indirect observations with other nodes in the network. The main disadvantages of second-hand observations include overhead, false report and collusion [30], [28].

Buchegger et al. [7] proposed the CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeTworks) protocol to encourage the node cooperation and punish misbehaving nodes. CONFIDANT has four components in each node: a monitor, a reputation system, a trust manager, and a path manager. The monitor is used to observe and identify abnormal routing behaviors. The reputation system calculates the reputation for each node in accordance with its observed behaviors. The trust manager exchanges alerts with other trust managers regarding node misbehaviors. The path manager maintains path rankings, and properly responses to various routing messages. A possible drawback of CONFIDANT is that an attacker may intentionally spread false alerts to other nodes that a node is misbehaving while it is actually a well-behaved node. Therefore, it is important for a node in CONFIDANT to validate an alert it receives before it accepts the alert.

Michiardi et al. [20] presented a mechanism with the name CORE to identify selfish nodes, and then compel them to cooperate in the following routing activities. Like CONFIDANT, CORE uses both a surveillance system and a reputation system to observe and evaluate node behaviors. Nevertheless, while CONFIDANT allows nodes exchange both positive and negative observations of their neighbors, only positive observations are exchanged among the nodes in CORE. In this way, malicious nodes cannot spread fake charges to frame the well-behaved nodes, and consequently avoid denial of service attacks toward the well-behaved nodes. The reputation system maintains reputations for each node, and the reputations are adjusted upon receiving of new evidences. Since selfish nodes reject to cooperate in some cases, their reputations are lower than other nodes. To encourage node cooperation and punish selfishness, if a node with low reputation sends a routing request, then the request will be ignored and the bad reputation node cannot use the network.

Patwardhan et al. [31] studied an approach in which the reputation of a node is determined by data validation. In this approach, a few nodes, which are named as Anchor nodes here, are assumed to be pre-authenticated, and thus the data they provide are regarded as trustworthy. Data can be validated by either agreement among peers or direct communication with an anchor node. Malicious node can be identified if the data they present is invalidated by the validation algorithm.

Ren et al. [32] proposed a node evaluation scheme in which each node evaluates the trustworthiness of its neighbors with the assistance of trustworthy neighboring nodes. Second-hand observations may be obtained from only a subset of the node's neighbors and these selected neighbors are regarded as trustworthy sources with respect to the opinions toward other nodes.

In our previous work [9], [10], [11], [12], we addressed the need for node behavior assessment by deploying a simple yet effective trust management scheme. The trustworthiness of each node is represented by one scalar

value and all the observation results are used to derive this single scalar trustworthiness.

The main differences between our trust management scheme and other related trust management methods are as follows:

- Introduction of *multi-dimensional* trust instead of a single scalar trust.
- Distinct evolution paces for different dimensions of the trust according to the basic features of the misbehaviors that are associated to the dimension of trust.

## 4 System Models

In this section, we present the system and adversary models that are used in this paper.

### 4.1 System Model

In this paper, we view a MANET as a set $\Delta$ of $N$ mobile nodes, that is, $|\Delta| = N$. The network size $N$ may dynamically change when nodes join, leave, or experience a variety of failures (such as communication interruption and exhausted battery power). Every node $A \in \Delta$ has a unique ID, which may generally be represented by its network-layer address.

The term *node* is defined as a system entity in MANETs that is capable of observing the behaviors of other nodes within its radio transmission range, and exchanging these observations with other nodes within its radio transmission range. We define a *neighbor* of a node $A$ as a node that resides within $A$'s radio transmission range. The type of abnormal behaviors that each node observes can be defined by the nodes themselves as long as all the nodes observe the same set of abnormal behaviors.

While a node observes the abnormal behaviors that its neighbors conduct, it also keeps track of the total amount of incoming packets it has observed for each neighbor. When a node needs to summarize its observation and thereby form its local view of misbehaving nodes, it will calculate the rate of abnormal behaviors over the overall behaviors it has observed for the node. For instance, if all the nodes choose to observe the behaviors of packet drop, modification and misroute, then *packet drop rate* (PDR), *packet modification rate* (PMOR) and *packet misroute rate* (PMIR) can be defined as follows, respectively.

$$PDR = \frac{Number\ of\ Packet\ Dropped}{Total\ Number\ of\ Incoming\ Packets}$$

$$PMOR = \frac{Number\ of\ Packet\ Modified}{Total\ Number\ of\ Incoming\ Packets}$$

$$PMIR = \frac{Number\ of\ Packet\ Misrouted}{Total\ Number\ of\ Incoming\ Packets}$$

We define the *trustworthiness* of a node $N_k$ as a vector $\Theta_k = (\theta_k^{(1)}, \theta_k^{(2)}, \cdots, \theta_k^{(n)})$, in which $\theta_k^{(i)}$ stands for the $i$-th dimension of the trustworthiness for the node $N_k$. Each dimension of the trustworthiness $\theta_k^{(i)}$ corresponds to one or a certain category of behavior(s) $B_k^{(i)}$ (such as packet forwarding or true opinion spreading), and $\theta_k^{(i)}$ can properly reflect the probability with which the node will conduct $B_k^{(i)}$ in an appropriate manner. $\theta_k^{(i)}$ can be assigned any real value in the range of $[0, 1]$, i.e., $\forall i \in \{1, 2, \cdots, n\}$, $\theta_k^{(i)} \in [0, 1]$. The higher the value of $\theta_k^{(i)}$, the node $N_k$ is more likely to conduct $B_k^{(i)}$ in a proper manner.

Each dimension of the trustworthiness $\theta_k^{(i)}$ for the node $N_k$ is defined as a function of the misbehaviors $M_k^{(i)}$ that are related to $B_k^{(i)}$ and have been observed by the neighbors of the node $N_k$. Different dimensions of the trustworthiness may correspond to different types of functions, and the selection of different functions should coincide with the basic features of $M_k^{(i)}$, such as severity of the outcome, occurrence frequency, and context in which they occur.

### 4.2 Adversary Model

In MANETs, each node may choose to either cooperate with other nodes as well as follow all the network protocols, or their behaviors noticeably diverge from the behaviors of other nodes. Despite that the divergence can be caused by both malicious intents and out of ignorance, those abnormal behaviors are both regarded as misbehaviors. A nodes that conducts some of all of those misbehaviors are regarded as an *adversary*.

Our goal in this paper is to contrive a sound misbehavior detection and trust management framework to cope with node misbehaviors in MANETs. Therefore, we assume that the adversary aims to disrupt network operations by conducting a variety of misbehaviors at various layers, such as malicious flooding of the Request-To-Send (RTS) frames in the MAC layer, dropping, modification, and misroute to the packets in the network layer, and deliberate propagation of fake observations regarding the behaviors of other nodes in the application layer. While it is also important to mitigate the attacks that generally target key management protocols in MANETs (such as the man-in-the-middle attack discussed in [33]), these key management attacks are beyond the scope of this paper.

We further assume that the adversary is able to mix its misbehaviors at any ratio if it choose to conduct multiple misbehaviors at the same time period. In addition, the adversary may alter the ratio of each misbehavior over time, and it can carry out the set of misbehaviors for any arbitrary length of time. For instance, an adversary $A$ may determine at time $t_1$ that it should equally conduct the four types of misbehaviors (i.e., RTS flooding, packet dropping, packet modification, and packet misroute); while at time $t_2$, $A$ changes its attack model to solely perform RTS flooding attack.

## 5 The SMART Framework

In this section, we present the SVM-based misbehavior detection and trust management framework (SMART)

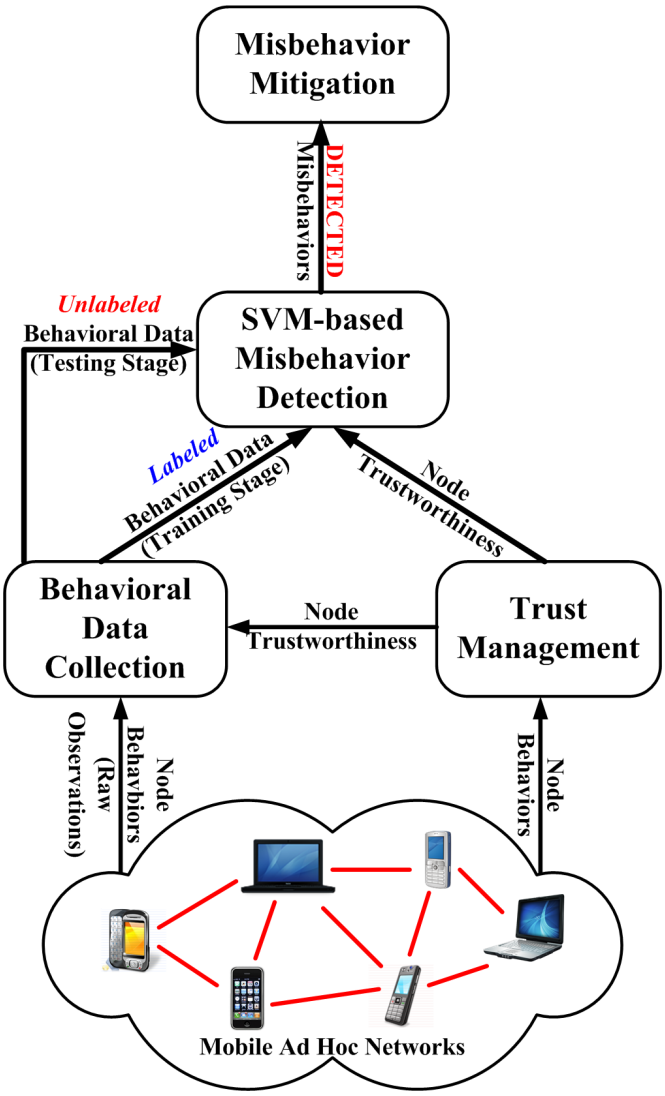



Fig. 2. The SMART Framework

in details. The goal of the framework is to recognize and mitigate misbehaviors by means of misbehavior detection and trustworthiness assessment for the nodes.

### 5.1 Framework Overview

In SMART framework, there are four main functional modules, namely *Behavioral Data Collection (BDC)*, *Trust Management (TM)*, *Misbehavior Detection (MD)*, and *Misbehavior Mitigation (MM)*. Figure 2 depicts the SMART framework.

### 5.2 Behavioral Data Collection

Behavioral Data Collection module is responsible for the collection of node behaviors and formation of behavioral dataset. We use network simulations to generate behavioral dataset and train a SVM classifier. Because the adversaries and their misbehaviors are pre-defined in these simulations, the behavioral data are collected and then labeled according to the ground truth regarding

| $Node\ ID$ | $PDR$ | $PMR$ | $RTS$ |
|---|---|---|---|
| 1 | 90% | 10% | 0 |
| 2 | 2% | 0 | 0 |
| 3 | 30% | 60% | 10% |
| 4 | 5% | 0 | 0 |
| 5 | 10% | 0 | 90% |
| ... | ... | ... | ... |

TABLE 1
An example training dataset.

adversaries. The trained SVM classifier can then be distributed and deployed to mobile devices. An example training dataset is shown in Table 1.

Here, we create a $m$-dimensional feature vector for each node. In the example shown in Table 1, $m = 3$.

Unlike the training stage, the set of misbehaving nodes is NOT known ahead of time in the testing stage. Therefore, the BDC module simply collects the local observations and forms a local view of node behaviors as the input for the testing stage of the MD module. Then, the *unlabeled* behavioral dataset is used by the MD module to detect the misbehavior nodes.

### 5.3 Multi-dimensional Trust Management

In the multi-dimensional trust management framework, the trustworthiness of a node $N_k$ is assessed in three dimensions, i.e., $\Theta_k = (\theta_k^{(1)}, \theta_k^{(2)}, \theta_k^{(3)})$. The three dimensions $\theta_k^{(1)}$, $\theta_k^{(2)}$, and $\theta_k^{(3)}$ are called *Collaboration Trust* (COT), *Behavioral Trust* (BET), and *Reference Trust* (RET), respectively. The three dimensions of trustworthiness are demonstrated in Figure 3.

From Figure 3 we find that COT ($\theta_k^{(1)}$) is determined by how collaborative a node $N_k$ would be when it is asked to participate in some network activities such as route discovery and packet forwarding. BET ($\theta_k^{(2)}$) is derived by the amount of abnormal behavior that $N_k$ has conducted, including packet modification, packet misroute or RTS flooding attack. RET ($\theta_k^{(3)}$) is generally computed based on the correctness of the observation results that $N_k$ spreads. For instance, if $N_k$ has been witnessed repeatedly sending fake observations to its neighbors, then $\theta_k^{(3)}$ should be assigned a rather low value. In this way, other nodes can properly interpret or even ignore the observations offered by $N_k$ because $\theta_k^{(3)}$ is used as the weight for $N_k$ when those observations are integrated to the local views of those nodes themselves.

Note that different categories of misbehaviors may occur in different contexts. Moreover, the consequences that these misbehaviors lead to can range significantly from loss of one packet to a benign node being framed by fake opinions and consequently trapped into denial of service. However, most of the existing trust management schemes do not take these factors into consideration, and they generally punish all the misbehaviors on a uniform

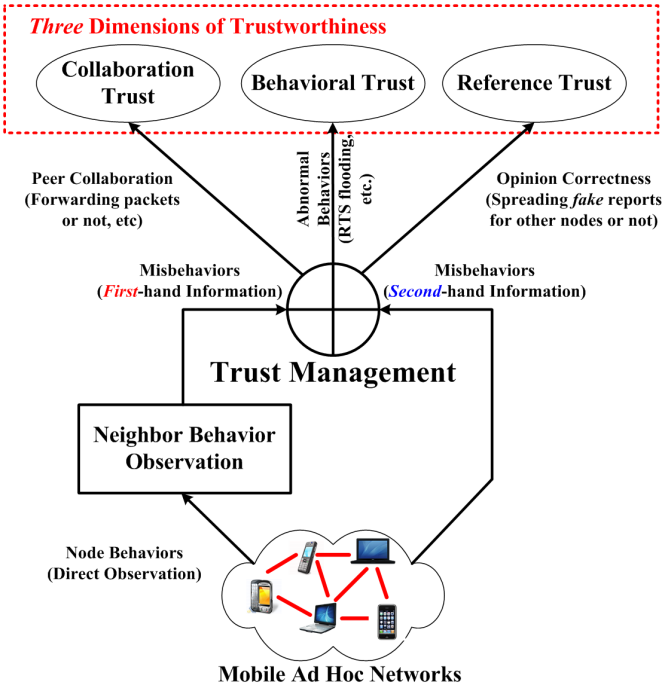



Fig. 3. The Three Dimensions of Trustworthiness

scale. To better take the nature of misbehaviors into account, we have developed an *adaptive* trustworthiness evolution model for different dimensions of trustworthiness, or even for the same dimension in different contexts.

Let us take the three dimensions of trustworthiness that we define as an example. Given that packet dropping may be caused by both malicious intent and environmental factors such as overflown buffer and exhausted battery, *Collaboration Trust* ($\theta_k^{(1)}$) should be reduced at a lower rate when compared to *Behavioral Trust* ($\theta_k^{(2)}$) because both packet modification and flooding of RTS frames can never be owed to environmental factors. Similarly, it is really harmful to spread fake observations in MANETs because the fake observations can cause massive chaos when the nodes attempt to distinguish between trustworthy neighbors and untrustworthy ones from their behaviors. As a result, *Reference Trust* ($\theta_k^{(3)}$) should decrease at the highest rate when compared to both COT and BET. Based on these arguments, we may utilize *logarithmic model*, *linear model*, and *exponential model* for $\theta_k^{(1)}$, $\theta_k^{(2)}$, and $\theta_k^{(3)}$, respectively. In other words, the following formulas should hold for the trustworthiness of the node $N_k$.

$$\theta_k^{(1)} \propto (a_1 * \log M_k^{(1)} + b_1)\ ,\ a_1, b_1 \in \mathbb{Q}$$

$$\theta_k^{(2)} \propto (a_2 * M_k^{(2)} + b_2)\ ,\ a_2, b_2 \in \mathbb{Q}$$

$$\theta_k^{(3)} \propto (a_3 * c_3^{M_k^{(3)}} + b_3)\ ,\ a_3, b_3, c_3 \in \mathbb{Q}$$

Not only can the trust evolution model differs for different dimensions of trustworthiness, it can also alter for the same dimension in different context. For example,

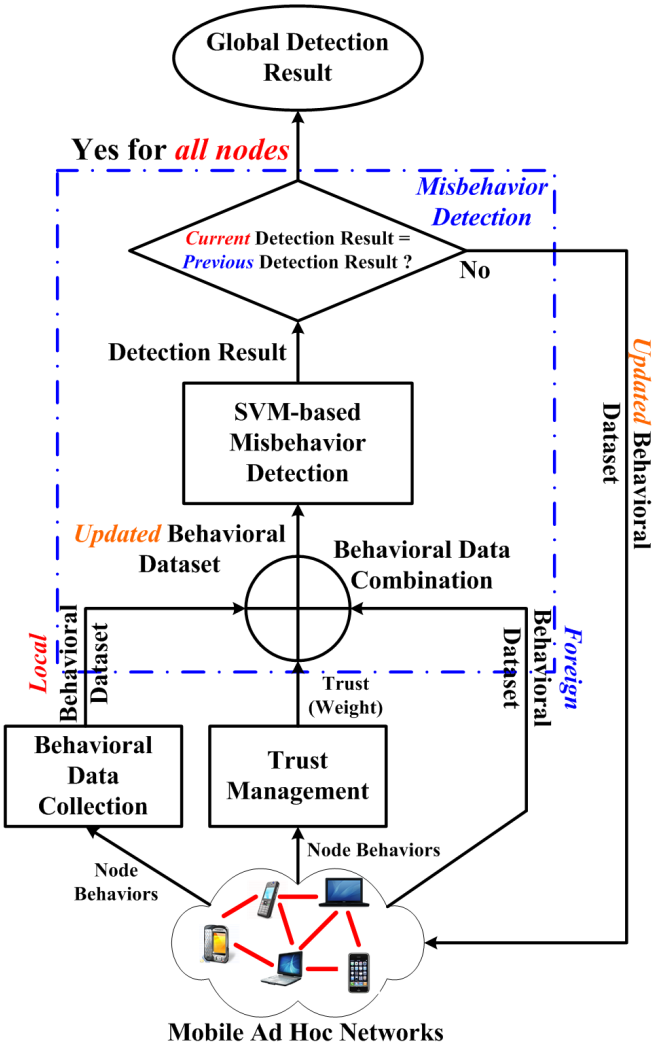



Fig. 4. SVM-based Misbehavior Detection

because packet dropping may be caused by both malicious intent and environmental factors, we can collect the context in which packet dropping occurs. If we infer from the context that it is caused by environmental factors, then we can use *logarithmic model* for $\theta_k^{(1)}$. In contrast, if we decide that the packet dropping is the outcome of malicious intent, then we may use *linear model* for $\theta_k^{(1)}$ to speed up the punishment.

### 5.4 SVM-based Misbehavior Detection

In the SMART framework, Support Vector Machine (SVM) technique is used to distinguish misbehaving nodes from well-behaved nodes. Figure 4 shows the diagram of the MD module in the detection stage.

From Figure 4 we may find that there are two key operations in the MD module, namely *Behavioral Data Combination* and *Misbehavior Detection*. Each node initially derives a preliminary view of misbehaving nodes based on the local behavioral data observed by itself, and the local behavioral data is exchanged among its neighbors at the same time. Once a node receives for-

| $Parameter$ | $Value$ |
|---|---|
| Simulation area | $150m \times 150m, 300m \times 300m$ $450m \times 450m, 600m \times 600m$ |
| Num. of nodes | 50, 100, 150, 200 |
| Transmission range | $60m, 90m, 120m$ |
| Mobility pattern | *Random Waypoint* |
| Node motion speed | $5m, 10m, 20m$ |
| Num. of bad nodes | 5, 10, 20 |
| Simulation time | 900s |

TABLE 2
Simulation Parameters

eign behavioral data from its neighbors, it integrates those foreign behavioral data into its own behavioral data using the trustworthiness of those neighbors. The multiple pieces of behavioral data are fused together using the Dempster-Shafer Theory of evidence (DST) [34], which has been discussed in details in our previous work [10]. The updated behavioral data will then be fed into the SVM classifier, and an updated view of misbehaving nodes is obtained as the new output of the SVM classifier. Next, the updated view is compared with the previous view, and the updated behavioral data should be broadcast to all neighbors if the two views are not identical. When all the nodes find that there is not any change in their local views when they receive foreign behavioral data, the procedure terminates and all the nodes hold the same global view of misbehaving nodes.

## 6 Performance Study

In this section, we examine the performance of the SVM-based Misbehavior Detection and Trust Management framework (*SMART*), and its performance is compared to that of the baseline mechanism. The baseline mechanism that we choose here is the Multi-dimensional Trust management framework (*mTrust*) discussed in our prior work [12], and our prior work show that (*mTrust*) framework outperforms other well-known mechanisms [12].

### 6.1 Simulation Setup

We use GloMoSim 2.03 [35] as the simulation platform, and table 2 lists the parameters used in the simulation scenarios. Moreover, SVM$^{light}$ [36] is used to implement the SVM functionality in the Misbehavior Detection (MD) module.

We use four parameters to evaluate the correctness and efficiency of our SMART framework: *Precision*, *Recall*, *Communication Overhead* (CO), and *Convergence Time* (CT). CO and CT are defined as follows.

$$CO = \frac{Number\ of\ Packets\ for\ the\ Framework}{Total\ Number\ of\ Packets\ in\ the\ Network}$$

$$CT = Time\ taken\ to\ form\ a\ unique\ global\ view$$

| $Node\ ID$ | $Start$ | $End$ | $Drop$ | $Modify$ | $RTS$ |
|---|---|---|---|---|---|
| 1 | $0s$ | $900s$ | 80% | 20% | 0 |
| 2 | $0s$ | $900s$ | 0 | 50% | 50% |
| 3 | $0s$ | $900s$ | 30% | 30% | 40% |
| 4 | $0s$ | $900s$ | 20% | 10% | 70% |
| 5 | $0s$ | $900s$ | 10% | 0 | 90% |

TABLE 3
An Example of Misbehavior Setup for the Training Stage

Overall, the experiments can be divided into two phases: the training stage and the testing stage. In the training stage, there are totally 100 nodes in the simulated MANET, with 5 of them being misbehaving nodes and the rest 95 nodes being normal nodes. These 5 misbehaving nodes conduct a mixed set of various misbehaviors including packet dropping, packet modification, RTS flooding, and fake observation spreading, and the mixture rates for these misbehaviors may vary among the misbehaving nodes. However, the mixture rate for each misbehaving node is fixed throughout the training stage. Table 3 depicts an example setup of misbehaviors for our simulation. Note that *Start* and *End* denote the start time and end time for the specified set of misbehaviors respectively. Each value in the columns of *Drop*, *Modify* and *RTS* stands for the percentage of the corresponding misbehavior over the whole set of misbehaviors in the specified time range, and the percentages of these three misbehaviors sum up to 1. Just take the first entry as an example: 80% of misbehaviors conducted by node 1 will be packet dropping and 20% of that be packet modification during the time range between 0s and 900s. The "rumor spreading" misbehavior will be defined separately.

In the testing stage, we vary the number of misbehaving nodes by 5, 10 or 20 in different experimental scenarios, not only the mixture rates for different nodes vary, the mixture rate for the same misbehaving node conducts may also differ over time. In this way, we guarantee that the training behavioral data are significantly different from the testing behavioral data. In addition, some adversaries may alter their attack models so as to make their behaviors less deviated from the normal behaviors. We will further discuss different adversary models that are considered in our experiments in Section 6.2.2.

Each simulation scenario has 30 runs with distinct random seeds, which ensures a unique initial node placement for each run. The experimental results are the average values over the 30 runs.

### 6.2 Simulation Results

The performance of *SMART* is observed and compared to that of *mTrust* in a variety of simulation scenarios. The simulation results show that: (1) In general, *SMART* achieves a good performance in terms of correct detec-

tion of misbehaviors, quickly convergence to a unique global view of misbehaviors among all the nodes, and acceptable communication cost; (2) *SMART* outperforms *mTrust* especially in the scenarios in which there are some *insidious* adversaries that periodically alter their attack patterns; and (3) The utilization of the adaptive trust evolution model helps improve the overall performance of *SMART* when compared to the previous trust evolution model in which all the trustworthiness evolve at a uniform (generally linear) pace. The simulation results are presented in details below.

#### 6.2.1 Overall Performance of SMART

To evaluate the performance of the proposed *SMART* framework, we have observed the performance of *SMART* as well as that of *mTrust* in the following four scenarios: different number of nodes, different radio ranges, different percentage of misbehaving nodes, and different node motion speeds. Given that the simulation area remains unchanged in all these scenarios, we can observe from these scenarios the effect of node density, radio range, adversary percentage, and node mobility, respectively. Note that each of the misbehaving nodes mixes all the misbehaviors with a fixed rate in these experimental scenarios. In addition, there are 5 adversaries in the network except for the third scenario (i.e., different percentage of misbehaving nodes), in which the number of adversaries can be 5, 10 or 20. The simulation results are showed in the following Figure 5 and Figure 6.

We observe from Figure 5(a) that when the node density is increased, *SMART* yields an overall higher precision. Moreover,it outperforms *mTrust* in terms of precision score when the node density is identical for both of them. Figure 5(b) depicts that *SMART* achieves a better precision when the radio range for each node is wider. This is the case because the broader the radio range, the more likely it is for each node to exchange observations with other nodes, which may lead to a more accurate view of misbehaviors. As is shown by Figure 5(c), the precision of *mTrust* is remarkably degraded when there are a larger amount of adversaries. In contrast, *SMART* is far more resilient to a significantly larger amount of adversaries than *mTrust* in that the precision of *SMART* only has a slight decrease with a higher percentage of adversaries in the network. We can conclude from Figure 5(d) that the higher the node motion speed, the lower the precision will be for both *SMART* and *mTrust*. Nevertheless, *SMART* is still able to yield a high precision score when the nodes are moving really fast.

Figure 6(a) plots the recall score for both *SMART* and *mTrust* versus different number of nodes. We find from Figure 6(a) that *SMART* surpasses *mTrust* for the recall when the number of nodes are 50 and 100 in the network, and the recall scores for both of them are similar when there are 200 nodes in the network. We can draw similar conclusions from Figure 6(b), 6(c), and 6(d) that *SMART* can generally achieve a higher recall than *mTrust*.

| $Node\ ID$ | $Start$ | $End$ | $Drop$ | $Modify$ | $RTS$ |
|---|---|---|---|---|---|
| 16 | $10s$ | $180s$ | 90% | 10% | 0 |
| 33 | $100s$ | $400s$ | 0 | 30% | 70% |
| 34 | $50s$ | $200s$ | 40% | 30% | 30% |
| 44 | $400s$ | $660s$ | 0% | 50% | 50% |
| 45 | $350s$ | $600s$ | 20% | 0 | 80% |

TABLE 4
An Example of *Short-term* Adversary Model

| $Node\ ID$ | $Start$ | $End$ | $Drop$ | $Modify$ | $RTS$ |
|---|---|---|---|---|---|
| 16 | $0s$ | $200s$ | 10% | 70% | 20% |
| 16 | $200s$ | $400s$ | 50% | 0% | 50% |
| 16 | $400s$ | $900s$ | 90% | 10% | 0 |
| 33 | $0s$ | $400s$ | 0 | 40% | 60% |
| 33 | $400s$ | $900s$ | 30% | 30% | 40% |
| ... | ... | ... | ... | ... | ... |

TABLE 5
An Example of *Everchanging* Adversary Model

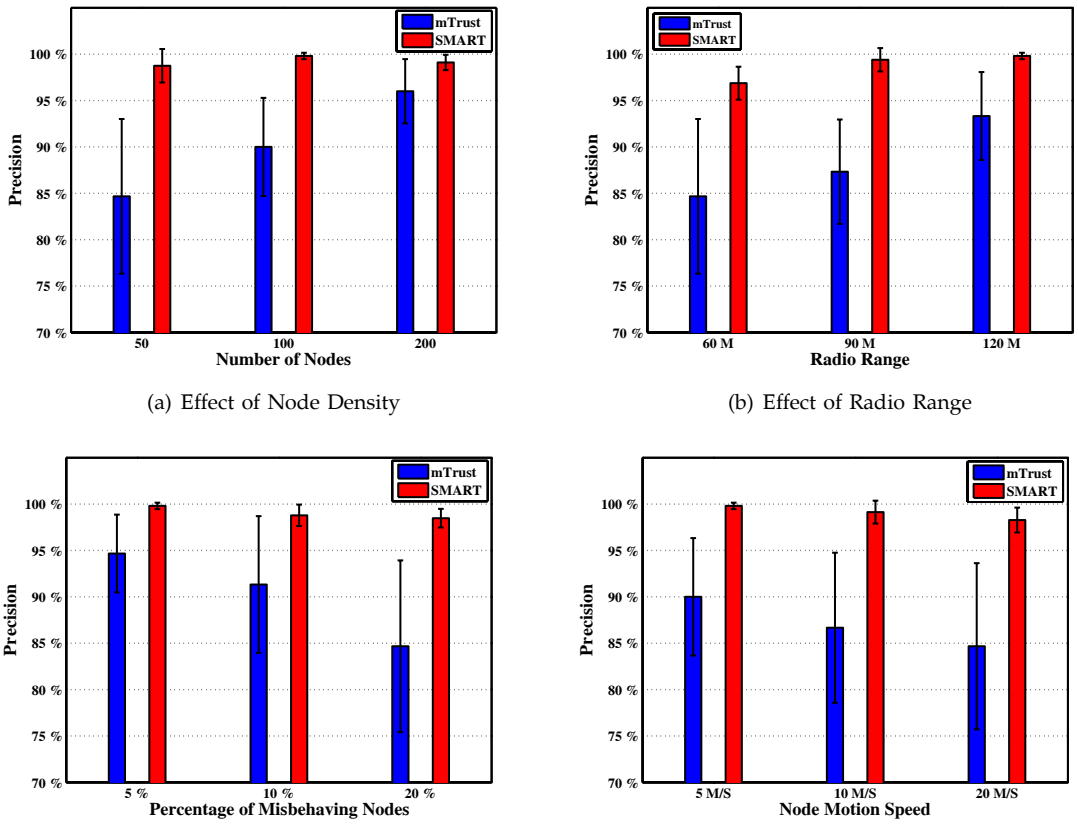
In addition to the simulation results discussed above, we also observe the performance of *SMART* and *mTrust* in terms of communication overhead and convergence time. The simulation results illustrate that *SMART* yields a good performance in that it converges in a short period of time with a small communication overhead.

#### 6.2.2 Effect of Changing Adversary Models

In this simulation scenario, the performance of *SMART* and *mTrust* are compared under different adversary models. As we discuss in Section 2, some *insidious* adversaries may choose to periodically alter their attack patterns so that their misbehaviors are harder to get detected. For example, an adversary can change the duration as well as the mixture rate of its misbehaviors from time to time, which makes its misbehaviors not so diverse from the normal behaviors. More specifically, we identify two special types of adversary model, namely *Short-term* and *Everchanging*, in our experiments. Table 4 and 5 show an example of *Short-term* and *Everchanging* adversary models, respectively.

We have deployed these two special adversary models in our experiments. To the best of our knowledge, the attack pattern of malicious nodes does not change in traditional mechanisms. Therefore, the *Comprehensive* adversary model is used as the baseline for comparison, in which each adversary may choose a different mixture rate of misbehaviors and keep this rate unchanged for the whole simulation time. The simulation results are shown in Figure 7.

From Figure 7(a) and Figure 7(b) we observe that *SMART* always exhibits a better performance over *mTrust* in terms of a higher precision as well as a higher recall. Figure 7(c) depicts that it generally takes less time for *SMART* than *mTrust* to reach a consistent global




(a) Effect of Node Density

(b) Effect of Radio Range

(c) Effect of Adversary Percentage

(d) Effect of Node Mobility

Fig. 5. Precision of *SMART* V.S. *mTrust*

view of misbehaving nodes. Moreover, we may find from Figure 7(d) that *SMART* usually introduces smaller communication overhead than *mTrust* does under all the three adversary models. Therefore, we can conclude from Figure 7 that when compared to *mTrust*, *SMART* always yields a better performance under all adversary models in that *SMART* achieves higher precision and recall scores in shorter time and with lower communication cost.

## 7 Conclusion

In this paper, a SVM-based Misbehavior Detection and Trust Management (*SMART*) framework is proposed and studied to identify misbehaviors and assess the trustworthiness of nodes.

The key contributions of our work are: (1) A novel SVM-based misbehavior detection method using outlier detection technique; (2) A multi-dimensional trust management framework in which the notion of trustworthiness is further classified into several dimensions so that each dimension is able to precisely indicate whether or not a node is trustworthy in terms of one specific behavior that it should conduct, such as cooperation, well-behaving, and honest; and (3) an adaptive trust evolution model by which each dimension of the trustworthiness can be adjusted according to the features of the misbehavior to which the dimension is related, such as severity of the outcome, frequency of occurrence, and context in which the misbehavior occurs.

Simulation results obtained from various scenarios have proven that the proposed SMART framework itself is resilient to various misbehaviors, and it can quickly converge to an accurate view of misbehaviors for each node in MANETs with an acceptable communication cost.

## References

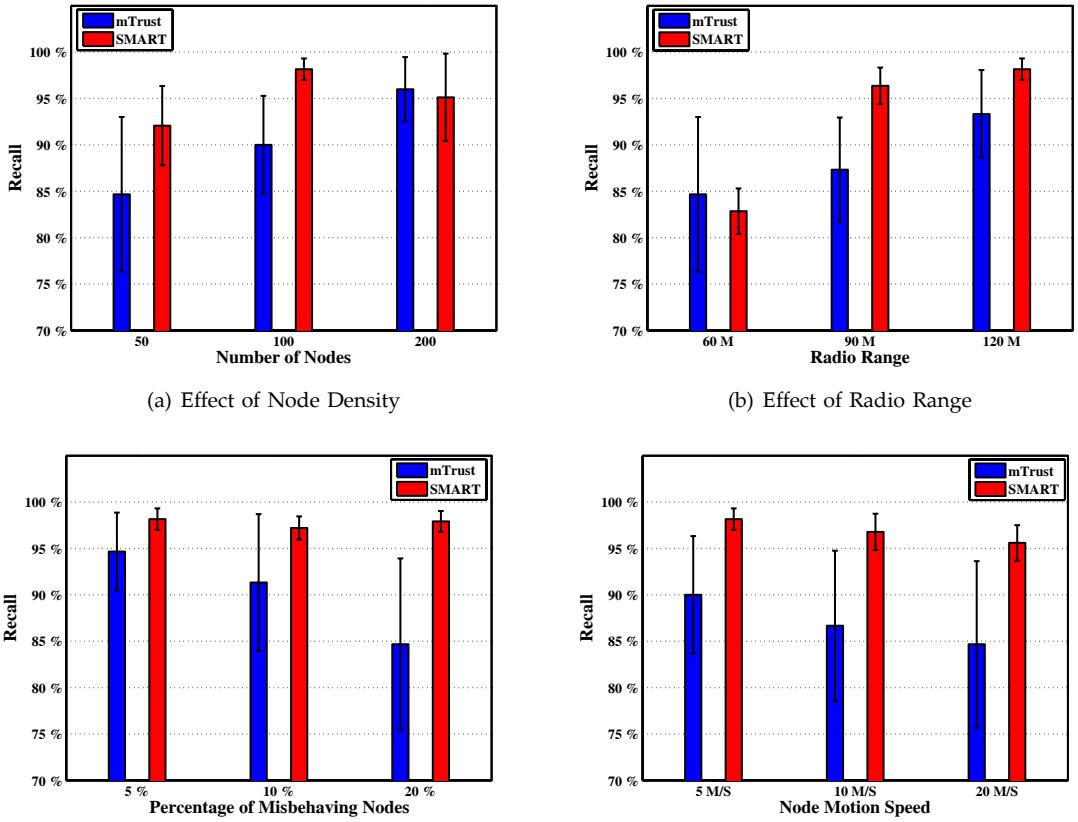
[1] W. Lu, W. K. Seah, E. W. Peh, and Y. Ge, "Communications support for disaster recovery operations using hybrid mobile ad-hoc networks," in *Proceedings of 32nd Annual IEEE Conference on Local Computer Networks (LCN 2007)*. IEEE Computer Society, 2007, pp. 763–770.

[2] R. Ramanathan and J. Redi, "A brief overview of ad hoc networks: challenges and directions," *IEEE Communications Magazine*, vol. 40, no. 5, pp. 20–22, August 2002.

[3] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in *Proceedings of MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004)*, 2004.

[4] Park, Soyoung and Zou, Cliff C., "Reliable Traffic Information Propagation in Vehicular Ad-Hoc Networks," in *Proceedings of 2008 IEEE Sarnoff Symposium*, 2008, pp. 1–6.

[5] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *MobiCom 00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 255–265.




(a) Effect of Node Density

(b) Effect of Radio Range

(c) Effect of Adversary Percentage

(d) Effect of Node Mobility

Fig. 6. Recall of *SMART* V.S. *mTrust*


[6] J. Parker, A. Patwardhan, and A. Joshi, "Cross-layer analysis for detecting wireless misbehavior," in *Proceedings of the Third IEEE Consumer Communications and Networking Conference, 2006. CCNC 2006.*, vol. 1. IEEE, Jan. 2006, pp. 6–9.

[7] S. Buchegger and J.-Y. Le Boudec, "Performance analysis of the confidant protocol," in *MobiHoc 02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2002, pp. 226–236.

[8] A. Patwardhan, J. Parker, A. Joshi, M. Iorga, and T. Karygiannis, "Secure routing and intrusion detection in ad hoc networks," in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, 2005. PerCom 2005.* IEEE, March 2005, pp. 191–199.

[9] W. Li, J. Parker, and A. Joshi, "Security through collaboration in manets," in *Proceedings of 4th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2008*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), vol. 10. Springer, 2008, pp. 696–714.

[10] W. Li and A. Joshi, "Outlier detection in ad hoc networks using dempster-shafer theory," in *Proceedings of the Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, 2009. MDM 09.* IEEE Computer Society, May 2009, pp. 112–121.

[11] W. Li, A. Joshi, and T. Finin, "Policy-based malicious peer detection in ad hoc networks," in *Proceedings of the International Conference on Computational Science and Engineering, 2009. CSE 09.*, vol. 3. IEEE Computer Society, Aug. 2009, pp. 76–82.

[12] W. Li, A. Joshi, and T. Finin, "Coping with node misbehaviors in ad hoc networks: A multi-dimensional trust management approach," in *Proceedings of the Eleventeenth International Conference on Mobile Data Management, 2010. MDM 10.* IEEE Computer Society, May 2010.

[13] W. Li, J. Parker, and A. Joshi, "Security through collaboration and trust in manets," *ACM/Springer Mobile Networks and Applications*, pp. 1–11, 2010.

[14] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *MobiCom 00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 275–283.

[15] G. Theodorakopoulos and J. S. Baras, "Trust evaluation in ad-hoc networks," in *WiSe 04: Proceedings of the 3rd ACM workshop on Wireless security*. New York, NY, USA: ACM, 2004, pp. 1–10.

[16] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas, "Robust cooperative trust establishment for manets," in *SASN 06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2006, pp. 23–34.

[17] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines : and other kernel-based learning methods*, 1st ed. Cambridge University Press, March 2000.

[18] S. Buchegger and J.-Y. Le Boudee, "Self-policing mobile ad hoc networks by reputation systems," *IEEE Communications Magazine*, vol. 43, no. 7, pp. 101–107, July 2005.

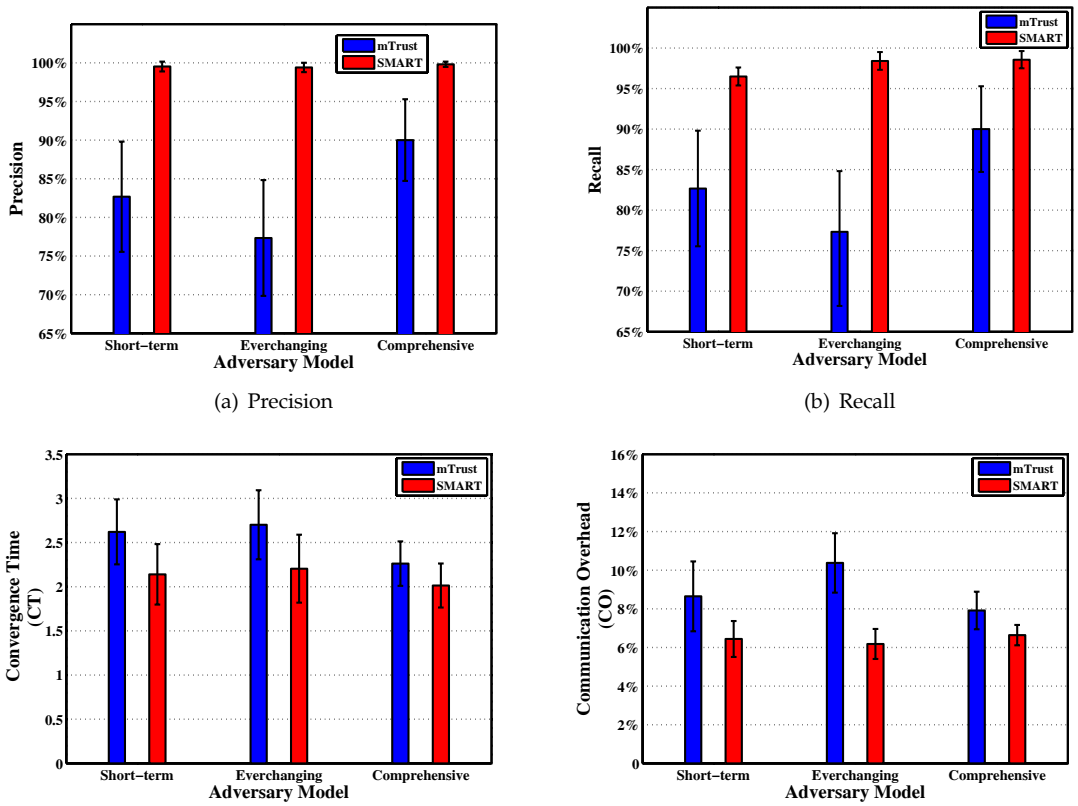
[19] P.-W. Yau and C. J. Mitchell, "Security vulnerabilities in ad hoc networks," in *Proceedings of the 7th International Symposium on Communication Theory and Applications*, 2003, pp. 99–104.

[20] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V., 2002, pp. 107–121.

[21] L. Zhou and Z. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, Nov/Dec 1999.

[22] H. Deng, Q.-A. Zeng, and D. Agrawal, "Svm-based intrusion detection system for wireless ad hoc networks," in *Proceedings of 2003 IEEE 58th Vehicular Technology Conference, 2003. VTC 2003-Fall.*, vol. 3, Oct. 2003, pp. 2147–2151.

[23] C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt, "A specification-based intrusion detection




(a) Precision (b) Recall

(c) Convergence Time (CT) (d) Communication Overhead (CO)

Fig. 7. Effect of Different Adversary Models on *SMART* and *mTrust*

system for aodv," in *SASN 03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2003, pp. 125–134.

[24] Y.-A. Huang and W. Lee, "A cooperative intrusion detection system for ad hoc networks," in *SASN 03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2003, pp. 135–147.

[25] L. Anderegg and S. Eidenbenz, "Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *MobiCom 03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003, pp. 245–259.

[26] Y. Xue and K. Nahrstedt, "Providing fault-tolerant ad hoc routing service in adversarial environments," *Wirel. Pers. Commun.*, vol. 29, no. 3-4, pp. 367–388, 2004.

[27] M. Kefayati, H. R. Rabiee, S. G. Miremadi, and A. Khonsari, "Misbehavior resilient multi-path data transmission in mobile ad-hoc networks," in *SASN 06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2006, pp. 91–100.

[28] S. Buchegger and J.-Y. L. Boudec, "The effect of rumor spreading in reputation systems for mobile ad-hoc networks," in *Proceedings of WiOpt 2003: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[29] S. Buchegger and J.-Y. L. Boudec, "A robust reputation system for mobile ad-hoc networks," in *Proceedings of P2PEcon*, 2003.

[30] Q. He, D. Wu, and P. Khosla, "Sori: a secure and objective reputation-based incentive scheme for ad-hoc networks," in *Proceedings of 2004 IEEE Wireless Communications and Networking Conference, WCNC 04.*, vol. 2, March 2004, pp. 825–830 Vol.2.

[31] A. Patwardhan, A. Joshi, T. Finin, and Y. Yesha, "A data intensive reputation management scheme for vehicular ad hoc networks," in *Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops, Mobiquitous 06.*, July 2006, pp. 1–8.

[32] Y. Ren and A. Boukerche, "Performance analysis of trust-based node evaluation schemes in wireless and mobile ad hoc networks," in *Proceedings of 2009 IEEE International Conference on-Communications, ICC 09.*, june 2009, pp. 1–5.

[33] B. Wu, J. Wu, E. B. Fernandez, M. Ilyas, and S. Magliveras, "Secure and efficient key management in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 30, no. 3, pp. 937–954, 2007.

[34] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ, USA: Princeton University Press, 1976.

[35] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," *ACM SIGSIM Simulation Digest*, vol. 28, no. 1, pp. 154–161, 1998.

[36] T. Joachims, *Making large-scale support vector machine learning practical*. Cambridge, MA, USA: MIT Press, 1999, pp. 169–184.