Data Collection, Dissemination, and Security in Vehicular Ad Hoc Network

by

Tong Zhou

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Krishnendu Chakrabarty, Supervisor

_____
Romit Roy Choudhury

_____
Kishor Trivedi

_____
Benjamin Lee

_____
Landon Cox

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University
2015

Abstract

Data Collection, Dissemination, and Security in Vehicular Ad Hoc Network

by

Tong Zhou

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Krishnendu Chakrabarty, Supervisor

_____
Romit Roy Choudhury

_____
Kishor Trivedi

_____
Benjamin Lee

_____
Landon Cox

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2015

Abstract

With fast-decreasing cost of electronic devices and the increasing use of mobile phones, vehicular ad hoc networks (VANETs) are emerging as a popular form of mobile ad hoc networks. VANETs are useful for supporting many applications that improve road safety and enhance driving convenience. Moreover, they also provide real-time data for traffic and travel management.

A VANET is composed of fast-moving mobile nodes (vehicles) that have intermittent and short contacts, fixed road-side units (RSUs) that overhear and broadcast to vehicles, and a central server. Vehicles move along roads, collect data and process them, and disseminate the data to other vehicles and RSUs. The central server aggregates data collected by vehicles, overviews traffic and road status, and generates keys and certificates when necessary. RSUs overhear the data sent from vehicles, broadcast road-side information to vehicles, and communicate to the central server via backhaul network.

With smartphones equipped on vehicles, many interesting research topics emerge, such as traffic-congestion detection and road-bump detection. After data are collected and processed, they are disseminated, such that other vehicles can collaboratively sense the road and traffic status. This motivates the need for data dissemination algorithms in a VANET. Due to the limited bandwidth and insufficient coverage

of 3G/4G networks, direct peer-to-peer communication between nodes is important.

Other major concerns in a VANET are security and privacy, since a malicious user can track vehicles, report false alarms, create undesirable traffic congestion, and illegally track vehicles. It is important to ensure the authenticity of messages propagated within VANETs, while protecting the identity and location privacy of vehicles that send messages.

This thesis addresses data collection, data processing, dissemination, and security. First, we estimate the location of vehicles in the scenario of weak/faded GPS signals by using the built-in sensors on smartphones. This method combines landmark recognition and Markov-chain predictions to localize a vehicle. Compared to the coarse-grained localization embedded in an Android smartphone using cellular and wifi signals, this method significantly improves accuracy. For data dissemination, we observe habitual mobility as well as deviations from habits, characterize their impact on latency, and exploit them through the Diverse Routing (DR) algorithm. Comparing to existing protocols, we show that DR leads to the least packet delay, especially when users deviate from expected behavior.

An important challenge for secure information dissemination in a VANET lies in Sybil attacks, where a single vehicle fakes multiple identities. We propose the Privacy-perserving Detection of Sybil Attack Protocl (P^2DAP) scheme to detect such Sybil attacks. The P^2DAP method does not require any vehicle in the network to disclose its identity, hence privacy is preserved at all times. Our results also quantify the inherent trade-off between security, i.e., the detection of Sybil attacks and detection latency, and the privacy provided to the vehicles in the network.

Due to the dependency of P^2DAP on RSUs, and the fact that RSUs are usually

semi-trusted in VANETs, we need an additional protection mechanism for RSUs. This observation motivates the Predistribution and Local-Collaboration-Based Authentication (PLCA) scheme, which combines Shamir secret sharing with neighborhood collaboration. In PLCA, the cost of changing the infrastructure is relatively low, and any compromise of RSUs can be quickly detected.

In summary, this thesis addresses problems that are relevant to various stages of the life-cycle of information in a VANET. The proposed solution handles data collection, data processing and information extraction, data dissemination, and security/privacy issues. Together, these adverses contribute to a secure and efficient environment for VANET, such that better driving experience and safety can be achieved.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Abbreviations

VANET	Vehicular ad hoc network. Mobile ad hoc network composed of vehicles equipped with short-range radios.
LTE	Long-term evolution, a wireless communication standard for high-speed data transmit in mobile phones and devices.
RSU	Road-side unit, the fixed and semi-trusted node placed along the road.
GPS	Global Positioning System.
SVM	Support vector machine.
HMM	Hidden Markov model.
DTN	Delay-tolerant network. Mobile ad hoc networks in which the nodes can hardly establish continuous connectivity.
WTD	UCSD Wireless topology discovery trace.
MITRM	MIT reality mining trace.
CDS	Connected dominating set.
CA	Certified authority, the trusted central server in a network that distributes certificates and keys to other nodes.
RSA	The public key cryptography developed by R. **R**ivest, A. **S**hamir and L. **A**dleman.
TPD	Tamper-proof device.

GS Group signature.

MAC 1. Message access control layer in the context of networking; 2. Message authentication code in the context of security.

Acknowledgements

First of all, I would like to thank my PhD advisor, Prof. Krishnendu Chakrabarty, for his continuous support of my Ph.D. research as well as his patience, passion, and rigorous academic attitude. He has helped me not only throughout my Ph.D. study, but also in my career outside the school.

Besides my advisor, my sincere thank also goes to Prof. Romit Roy Choudhury and Prof. Peng Ning, for their great help during my Ph.D. study.

I would like to thank the rest of my thesis committee: Prof. Kishor Trivedi, Prof. Benjamin Lee, and Prof. Landon Cox, for all their valuable suggestions and comments during the thesis writing.

I an grateful to my fellow labmates: Dr. Qing Duan and Dr. Fangming Ye, for their great help and discussion in my research. I would also like to thank Dr. An Liu for his inputs in my Ph.D. project.

Last but not least, I would like to thank my family – my husband Candong Cheng, for his support in my Ph.D. study.

1

Introduction

As the field of wireless networking has matured and developed, people no longer rely on desktop computers or wired phones for network services. With the use of mobile computing devices such as laptops, PDAs, and cell phones, network services are accessible in many places, such as taxis, airports, or building hallways.

With advances in wireless communication and sensing technologies in recent years, car manufacturers and the telecommunication industry have started to equip vehicles with wireless devices that allow them to communicate with each other, with roadside units, and with sensors that observe the surrounding environment, including the road condition, weather, and traffic status. These add-ons have greatly improved driving safety and enhanced the driver's experience.

These vehicles equipped with wireless devices form a Vehicular Ad Hoc Network (VANET) (Fujimoto et al., 2002; Kimura and Murakami, 1993). VANETs are being advocated as a method for increasing road safety and driving comfort as well as

facilitating traffic control (Franz et al., 2004; Kosch and Strassberger, 2004). These benefits are realized by collecting dynamic traffic information and sensing various physical quantities related to road conditions at a low cost but with high accuracy, thus achieving automatic and dynamic data collection and fusion in an intelligent transportation system. Moreover, widely used smartphones equipped with 3G/4G network interfaces and various sensors provide an even more powerful sensing platform for vehicles.

Various novel applications have emerged from this framework. For example, Waze (Bardin et al., 2009), a commonly-used smartphone localization and navigation app, is capable of observing street-level traffic conditions and dynamically adjusting the route of a car in transit. Another example is iOnRoad (Atsmon and Atsmon, 2011), which detects and monitors the distance between vehicles to maintain driving safety.

1.1 About This Thesis

A VANET must have ubiquity, scalable service, high availability, and efficient data dissemination. It also must ensure the security/privacy of both the disseminated data and the vehicles. Key challenges today that impede VANET deployment include the following: (i) coping with temporary GPS signal loss; (ii) efficient message forwarding; and (iii) vehicle privacy.

This dissertation addresses the above challenges. We show that it is feasible to (i) obtain the location of a vehicle even without a GPS signal, thus achieving ubiquitous localization service in a VANET; (ii) forward messages to a given destination node with close-to-optimal delivery delay and affordable overhead, even if the node moves randomly; (iii) provide a scalable scheme to address Sybil attacks while preserving

vehicle privacy; and (iv) detect RSU compromise in a distributed manner with low cost.

1.2 System Overview

In VANETs, there are three different types of nodes, listed as follows:

1. Central server (CSV). The server handles all data collected by other nodes and performs data mining. It also serves as the Certified Authority (CA) and distributes keys to other nodes.

2. Road-side unit (RSU). These nodes are fixed nodes with infrastructure support. They are a semi-trusted party with a direct connection to the central server and also have wireless devices that allow them to communicate with nearby vehicles.

3. Vehicles. These nodes are the major users of VANETs. They have high mobility and therefore intermittent wireless connections with each other; they also have a high requirement for privacy because it is not desirable for a vehicle to be tracked in a normal scenario.

Figure 1.1 illustrates how the vehicles move and communicate with various parties in the network.

1.2.1 Sensors on Vehicles

On a vehicle, there are various on-board sensors that sense a vehicle's motion and the surrounding environment. Figure 1.2 illustrates the driving and parking-assist sensors available for the Ford Addict car (Conner, 2011).

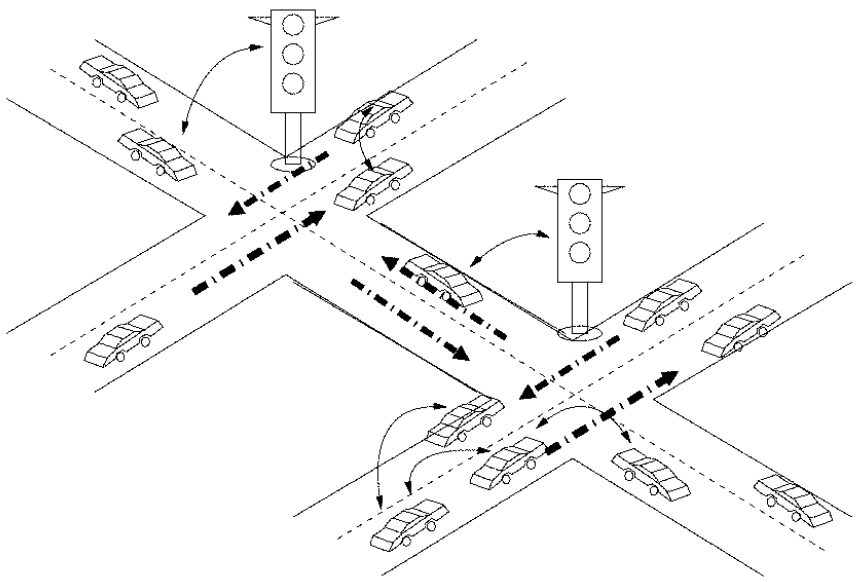


FIGURE 1.1: This figure illustrates how a VANET is formed. Vehicles move along roads in specific directions. There may also be road-side units (RSUs), such as traffic lights, that are fixed. Vehicles can communicate with each other and with RSUs.

Many of these on-board sensors can significantly improve driving safety. For example, speed sensors, wheel-speed sensors, and torque sensors can detect the vehicle's movements while the blind-spot detector and the parking-assist sensor can be used to detect obstacles around vehicles. On-board sensors can also warn drivers about approaching vehicles, speeding, and slippery road conditions.

With the increasing use of smartphones equipped with a rich set of sensors, vehicles are now able to detect road conditions with high accuracy. Many people dock smartphones on vehicles and use them as videocams and GPS units, but smartphones can serve as much more powerful tools. For example, the camera available on smartphones can detect speed-limit signs, stop signs, traffic lights, and the brake lights

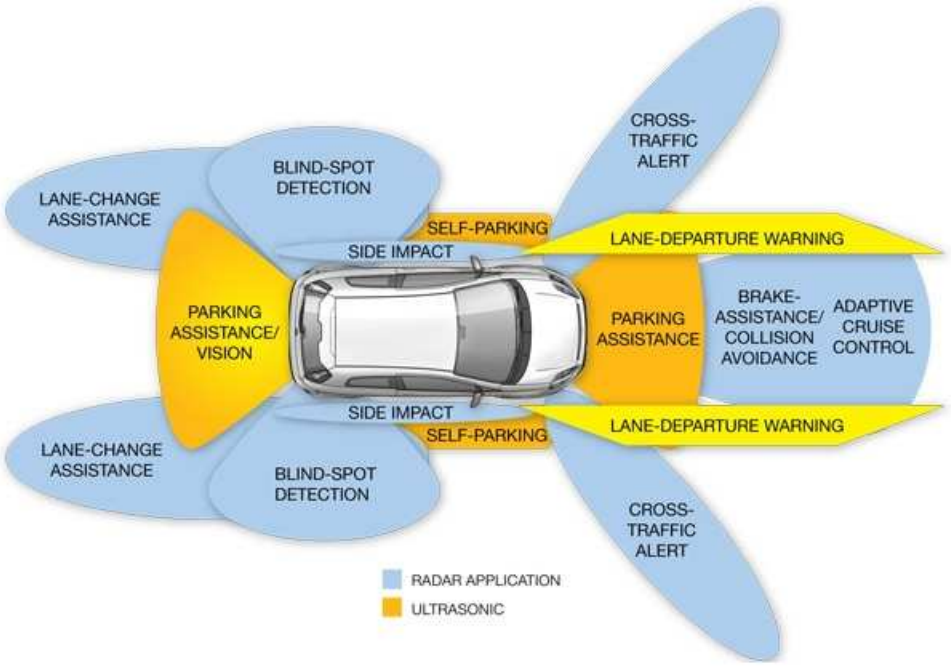


FIGURE 1.2: Onboard vehicle sensors and their functions (Conner, 2011).

of leading vehicles and then notify the driver for a safer driving experience. The microphone can detect the car horns of other vehicles, thus yielding information of the traffic status. The accelerometer and gyroscopes can detect unpaved roads and bumps and inform other vehicles of the road status. Moreover, smartphones provides access to 3G/4G networks, thus allowing full-mesh communication in VANET – vehicles are able to communicate with each other via Global System for Mobile Communications (GSM) or the 4G Long Term Evolution (LTE) network, albeit with limited bandwidth and limited amounts of data. When the data are processed and information is disseminated from one vehicle to the others, all of the vehicles on the road can collaborate and produce a much better driving experience.

Table 1.1: Comparison of various parameters of 802.11a and 802.11p.

Parameters	802.11a	802.11p
Data rate (Mbps)	6, 9, 12, 18, 24, 36, 48, 54	3, 4.5, 6, 9, 12, 18, 24, 27
Communication Radius (m)	35 – 120	100 – 400
Modulation	OFDM	OFDM
Frequency (GHz)	5	5.9
Bandwidth (MHz)	20	10

1.2.2 Inter-Vehicle Communication

In VANETs, the data collected on each vehicle is sent via the short-range radio that is available on the vehicle. Most recent research in the area of VANETs assumes that either the 802.11a (IEEE, 1999) or 802.11p (IEEE, 2010) standard has been used for vehicle-vehicle and vehicle-RSU communications (Woo and Han, 2010; Jansons and Barancevs, 2012; Abakar et al., 2010; Zemen et al., 2012). The 802.11a standard is an amendment to the 802.11 wireless local area network (WLAN) protocol that specifies the requirements for orthogonal frequency division multiplexing (OFDM) modulation. The 802.11p, or Wireless Access in Vehicular Environments (WAVE) protocol, is an amendment to 802.11 that provides enhancements to the physical (PHY) and medium access control (MAC) layers for the purpose of vehicular communications. A significant amount of research has been conducted on analyzing the performance of 802.11p (Jiang et al., 2006; Cheng et al., 2007; Jiang and Delgrossi, 2008; Yin et al., 2014) and providing various improvements such as congestion control and collision avoidance (Jiang et al., 2006; Ma et al., 2012).

A comparison between 802.11a and 802.11p discussed in the above papers is shown in Table 1.1.

Considering that there have been sufficient research on the PHY and MAC layers in a VANET, in this dissertation, we ignore all of the factors in these two layers that can possibly degrade either the packet delivery ratio or the delay. We assume that packet collisions and faded signals between vehicles and RSUs are well-handled. Moreover, we assume that the vehicle radio has a communication range of 100 – 200 meters. Assuming a typical driving speed of 40 – 110 km/h, the lifetime of an established connection between a vehicle and an RSU is estimated to be 5 – 20 seconds. On the other hand, for two vehicles that are in close proximity and driving on a road in the same direction, the connection can last for minutes. Such short and unstable connections give rise to several interesting research problems related to VANET data dissemination, which are listed as below.

1.3 Related Prior Work and Shortcomings

VANETs have attracted considerable attention in the past few years, and a large amount of research has been devoted to improving the driving experience and driving safety through various techniques such as data collection, data processing, data dissemination, and security.

1.3.1 Vehicle Localization

An important piece of information required in a VANET is the vehicle location, and thus localization has attracted significant attention in the research community. In outdoor scenarios, GPS is the technology that is most commonly used to provide accurate positioning for various applications, such as trace and tracking in the wild (Brown and Sturza, 1993) (Berard et al., 1996) and environmental

monitoring (Weimerskirch et al., 2002). However, the GPS signal can be easily disturbed or jammed, leaving many regions with no GPS signals. For example, (Shagimuratov et al., 2007; Basu et al., 2001; Tsugawa et al., 2003) have pointed out that geomagnetic storms can cause severe GPS signal fluctuations, resulting in signal loss or large errors in localization. Dense high-rise buildings on the ground can also block GPS signals. In (Melgard et al., 1994), researchers have studied the degradation of the GPS signal in urban areas caused by surrounding buildings.

Therefore, researchers have recently focused on vehicle localization without GPS (Bo et al., 2013) (Wu and Ranganathan, 2013). For example, (Bo et al., 2013) targets an urban environment where the GPS signal is intermittent and there are small areas of "shadows" in the GPS signal (100 – 200m) on the streets. Smartphones are trained outside the shadow regions, where the GPS signal is assumed to be strong, and then used to predict the location inside the shadow region, where the GPS signal is weak. Because the time-duration difference between training and predicting is short, this method can achieve high prediction accuracy. However, it cannot handle a large shadow region.

In (Wu and Ranganathan, 2013), researchers proposed to use the built-in camera to capture street-views and use them to predict the vehicle's location. This method also offers a high localization accuracy. However, image processing is computationally expensive on a smartphone (Gammeter et al., 2010).

Another research direction is the use of a sequence of sensor observations to obtain the trace of a vehicle. (Han et al., 2012) describes ACComplice, which uses a map-matching algorithm on a sequence of accelerometer readings to deduce the vehicle's trace. With this method, the authors describe a privacy hole in existing

VANETs that shows how a smartphone's accelerometer reading can leak the owner's past locations. One drawback of ACComplice is that it relies only on accelerometer readings and does not have an error-correction mechanism. On most of the roads considered in the experiments reported in (Han et al., 2012), the predicted location is on a parallel street 1-2 blocks away from the actual location. Once the system makes a mistake, it has no chance to correct the error as long as the vehicle continues along on the road without any turns. Therefore, this algorithm cannot produce an effective localization tool.

1.3.2 Data Dissemination

One important and desirable attribute of a VANET is that it covers all of the available areas, which is known as ubiquity. One may argue that broad-coverage wireless networks can satisfy most of the requirements of mobile nodes and therefore a VANET that mainly relies on vehicle-to-vehicle communication or vehicle-to-RSU communication is not necessary. For example, cell phones can easily communicate with each other with a reasonably good data rate, and current 3G/4G networks already cover a wide area. However, we highlight the following scenarios where these solutions are not sufficient as a VANET substitute:

1. There are still areas today that are not covered by cellular networks, e.g., remote villages in the developing and under-developed world. Communicating via cellular network in these areas is not feasible. However, it is feasible to use vehicles that travel in these areas and transmit data in a VANET.

2. In cellular networks, delivering a large volume of data at a high rate is expensive. In contrast, in VANETs, data packets are transmitted via short-range

radio connections and piggybacked on vehicles; as a result, the data delivery cost is much lower than for a cellular network.

3. A VANET is more scalable than a cellular network. Cellular network coverage is highly reliant on the installation of cellular towers, while VANETs can extend to any area that vehicles drive to.

Moreover, a VANET can be synergistically combined with a cellular network. For example, vehicles carrying smartphones can use GSM to send/receive small-size control packets while using a WiFi connection to upload large files or inform other vehicles in real time.

Compared to traditional networks, VANETs provide two major benefits:

1. Low cost. VANETs can be easily supported with current wireless devices such as smartphones and laptops. As discussed above, many vehicle-based apps have been developed on smartphones.

2. High scalability. We can extend VANETs to a wider area simply by allowing the nodes to move in the larger space, or we can extend VANETs to more users simply by allowing new users to join. However, in traditional networks, much more hardware support, such as the installation of new cellular towers or communication cables, is needed.

The observation that *mobility can be an opportunity* (Grossglauser and Tse, 2002) has inspired a host of new ideas. Researchers have proposed a variety of routing protocols that opportunistically exploit user mobility to transport data from a source to a desired destination (LeBrun et al., 2005; Burgess et al., 2006; Naumov and Gross,

2007; Wisitpongphan et al., 2007; Zhao and Cao, 2008; Jeong et al., 2011; Jang et al., 2011; Ros et al., 2012). Many of the above protocols rely on the prediction of vehicle motion within a short time frame based on the location, speed, and direction of the vehicle (LeBrun et al., 2005; Naumov and Gross, 2007; Wisitpongphan et al., 2007; Zhao and Cao, 2008; Jeong et al., 2011; Ros et al., 2012). In the above works, vehicles identify the trajectory of their neighboring vehicles and relay the information packets to those neighbors that are likely to deliver the frame (e.g., driving towards the destination or being closer to the destination). In (Zhao and Cao, 2008), vehicle relaying is used to address the scenario where the WiFi signal between a vehicle and an RSU is too weak – a neighboring vehicle between the source vehicle and the destination RSU is chosen to forward the packet.

In sparse VANETs or in VANETs where more pedestrians are involved, it is difficult for vehicles to predict the delivery rate simply based on vehicular motion. In these situations, the statistics of encounters plays a more important role. In (Burgess et al., 2006), data forwarding methods are investigated in a VANET that is mainly composed of campus buses with occasional encounters and fixed schedules and routes. The past encounters of buses are studied and then used to predict the data delivery ratio for the future. While the delay for such a data transportation method can be high and data forwarding is opportunistic, studies have shown that carefully replicating the data among redundant transporters can considerably reduce this delay as well as improve the delivery ratio (Kihl et al., 2007; Zhang et al., 2007; Lin et al., 2008). Similar work has been done on pedestrian movement. In this case, the goal is to capture the *regularity* in human movements and utilize it to intelligently guide data to the destination (Daly and Haahr, 2007; Hui et al., 2008;

Gonzalez et al., 2008; Gao et al., 2009).

1.3.3 Vehicle Privacy

While designing a VANET system, it is important to address security and privacy concerns (Raya and Hubaux, 2007). The system needs to be robust against non-cooperating entities and should ideally be able to detect/punish them quickly. One straightforward method to ensure the authenticity of messages propagated in a VANET is to use public keys certified by a certification authority (CA), called "pseudonyms," to sign messages. In a VANET, the CA is usually assumed to be a centralized server. To prevent vehicles from being tracked by identifying the keys that are used, each vehicle can switch between multiple pseudonyms, which are difficult to correlate with one other. With this approach, it is difficult for an attacker to identify vehicles by examining the used keys. The above scheme has been proposed by many researchers (e.g., (Zarki et al., 2002), (Sampigethaya et al., 2005), (Choi et al., 2005), (Calandriello et al., 2007), (Rass et al., 2008)), and it has been shown to work efficiently.

Although the above method protects the privacy of the vehicles, it contains another security hole. Because it is difficult to determine whether two messages are from the same vehicle by examining their public keys, a malicious vehicle may pretend to be multiple vehicles (*a Sybil attack*) and then distribute false information. The deleterious effects of such attacks can cascade through the network. In (Parno and Perrig, 2005), this problem was tackled by preloading vehicles with one temporary pseudonym with a short lifetime. Vehicles are thus expected to obtain a new pseudonym from a trusted Road-Side Unit (RSU) (serving as the CA)

immediately before the first pseudonym expires. The idea is further extended in (Studer et al., 2009b), where the roads are divided into geographical regions. When a vehicle enters a new region, or its current pseudonym is about to expire, it securely updates the pseudonyms (named TACKs in (Studer et al., 2009b)) with the CA either via the RSU in that region or via a cellular network. The solutions in (Parno and Perrig, 2005) and (Studer et al., 2009b) are light-weight and secure in the sense that vehicles only hold one valid pseudonym at a time. However, in these schemes, it is critical that a vehicle has access to a CA when it needs to update its pseudonym. Without such an online infrastructure support, the vehicles are unable to obtain new pseudonyms and send signed messages. Also, if an attacker compromises an RSU, he can collude with a few malicious vehicles by issuing a huge number of certified pseudonyms to those vehicles. With those pseudonyms, the malicious vehicles can then pretend to be many vehicles and flood false messages through the VANET.

Another technique exploits directional antennae to identify the position/direction from which a message arrives (Golle et al., 2004). A car launching a Sybil attack is expected to be detected because all of the duplicate messages will come from the same position. However, in dense networks, localization errors can lead to frequent false positives. More importantly, a smart attacker may use directional antennae to mislead its neighbors about its direction.

Lu et al. proposed a Sybil-attack-resistant scheme (Lu et al., 2012) that addresses this problem from a different angle. In their work, the keys distributed to each vehicle are carefully generated using a bilinear pairing technique such that, once a vehicle signs identical messages using the different pseudonyms, other vehicles immediately

determine that the two messages are from the same vehicle; on the other hand, signing different messages using different pseudonyms will not affect the privacy of the vehicles. This is an elegant scheme that can detect Sybil attacks in a distributed manner without introducing communication overhead. However, this solution only works for Sybil attacks that sign identical messages. An attacker can beat this solution by introducing very slight differences in the messages to be broadcast, e.g., make small changes to the timestamp or location.

1.3.4 Road-Side Unit Security

In many services that VANETs provide, such as those described in (Raya and Hubaux, 2007) and (Studer et al., 2009a), RSUs play an important role in terms of packet routing and security, but they are vulnerable to being compromised (Raya and Hubaux, 2007; Studer et al., 2009a; Kim et al., 2010).

RSU compromise can cause severe security holes in a VANET. In a hostile environment, an attacker that compromises an RSU may use it to send faked messages such that the vehicles are misinformed and/or make incorrect decisions. It may also send faked information to the CA and revoke benign vehicles, thus carrying out a Denial-of-Service (DoS) attack. It is expensive to deploy an RSU, and the RSU node can easily become the compromising target of the attackers if it is not adequately protected. In the case of an RSU compromise, the entire VANET may crash or malfunction.

In (Raya and Hubaux, 2007), it is assumed that RSUs can be compromised but it is quickly detected. In (Zhou et al., 2011), RSUs are assumed to be directly connected to the CA, which has some method to quickly detect a compromise. However,

it is not emphasized how the detection is carried out. In fact, very little work has been reported on how RSU compromise can be detected.

Handling Compromised RSUs: Vehicles as Watchers

In (Yang et al., 2006), researchers propose to use vehicles as RSU watchers. When a nearby vehicle determines that a particular RSU is sending false messages, it relays the message to other RSUs using the signatures of the malicious RSU. On receiving a sufficiently large number of such false messages from a nearby RSU, the uncompromised RSU informs the CA by forwarding all the messages sent by the vehicles. The CA can then use the SCAN scheme (Yang et al., 2006) to revoke the RSU. A key distribution mechanism for a VANET is discussed in (Hao et al., 2008), where researchers proposed a similar scheme for detecting RSU compromise. On seeing misbehaving RSUs/vehicles, vehicles can directly send an accusation message to the CA. The CA then decides how to handle the malicious node and revokes the node if necessary.

In (Ruj et al., 2011), compromised vehicles are voted out by other nearby vehicles. While this method can work well with vehicles, it may not fit the scenario of RSUs, which are more sparsely distributed and may not always have enough surrounding nodes to vote them out.

In (Lin et al., 2013), a node compromise and revoke model is discussed. In this paper, researchers noted the presence of connected dominating sets (CDS) in a VANET. In this scenario, a malicious node tries to pass malware to the vehicles in a CDS such that the number of affected vehicles is maximized. However, there is a defender node in a VANET that recovers the compromised node and assigns new keys to it.

The defender also prioritizes the recovery of the vehicles in a CDS such that it can efficiently prevent malware from propagating further.

In all of the methods discussed above, the researchers did not directly focus on the issue of RSU compromise, although methods for handing compromised RSUs are briefly discussed. In general, these methods expect nearby vehicles to detect and report compromised RSUs. This approach can expose more security vulnerability as follows:

1. Vehicles represent an untrusted party in a VANET. A malicious vehicle or several colluding vehicles can intentionally accuse a benign RSU and disable it.

2. If RSUs are sparsely distributed, it may be a long time before a vehicle reports a misbehaving RSU.

Tamper-Proof RSU

In (Raya and Hubaux, 2007; Zhang et al., 2008), researchers suggested the use of tamper-proof devices (TPDs) in VANETs. A TPD can effectively prevent an attacker from obtaining the private keys of an RSU and using it to send bogus messages. On the other hand, even if the RSU is tamper-proof, attackers can still use the RSU to send false messages, not by compromising its private keys but rather by compromising its environment. Although tamper-proof devices can be used to prevent attackers from compromising the keys or certificates stored in RSUs, they can still carry out attacks by forcing the RSU send false messages. One possible scenario in which an RSU can be fooled is described in (Mohan et al., 2008), where vehicle honks are

used to detect traffic congestion on the road. If this method is used on an RSU, an attacker can easily mislead an RSU by replaying vehicle honks near the RSU or blocking its built-in microphone, which causes the RSU to broadcast false messages.

Resisting RSU Compromise

Recently, there has been research on VANET security schemes that are resistant to RSU compromise. In (Ruan et al., 2011), a key management scheme based on the threshold ElGamal system (Elgamal, 1985) is proposed. In this system, each key is split and distributed to multiple RSUs that use the ElGamal key-sharing mechanism. If an RSU is compromised and then detected, other RSUs will revoke it by refusing to contribute their key shares. (Hao et al., 2011) proposes a key management scheme to specifically handle the compromised RSUs that collude with malicious vehicles, i.e., in a scenario where malicious vehicles send false messages and RSUs provide the keys they need. In this work, the false information sent from malicious vehicles is assumed to be detected and then reported to the CA by many benign users. The CA then verifies the reports by determining the signatures in the reports the identities of both the malicious vehicles and the compromised RSUs are then deduced. After that, the compromised RSUs can be revoked.

1.4 Motivation and Thesis Research Topics

Research topics in VANETs range from data collection to security during data dissemination. Previous work in these areas as discussed above reveal many interesting challenges.

In localization without GPS, most methods rely on the assumptions that GPS

signals are intermittent and dead-reckoning can be applied to an area without GPS. However, in some scenarios, the GPS signals can be severely impaired over a large area. For example, according to the study in (Tsugawa et al., 2003), a geomagnetic storm can impact an area range from tens to hundreds of kilometers. Thus, a localization method that can van be used in place of GPS is needed. In (Han et al., 2012), a GPS-free localization system ACComplice is proposed, which uses only accelerometer readings to predict a vehicle's location. However, it does not have any error correction mechanism, which motivates the need for another GPS-free localization system. To cover a large area with severely impacted or disturbed GPS signals, we propose a localization system that predicts a vehicle's location in real-time without consulting GPS readings along the entire path.

After the data is collected and processed, the data must be disseminated. Most of the work in this area focuses on vehicle-to-vehicle communications; this approach exploits vehicle speed and short encounter times to ensure fast data delivery. In recent years, however, there has also been interest in exploiting the role of pedestrians in a VANET. Vehicle-to-pedestrian communications have also been investigated in the operation of a VANET (Hwang et al., 2011; Wang et al., 2012). With smartphones, drivers on vehicles can easily talk to each other when they are not driving; vehicles on the road can also remind pedestrians of coming vehicles. In (Song et al., 2013), researchers note that traffic lights and pedestrians crossing the road can increase the carrying delay of disseminated data through waiting time, thus introducing holes in a VANET. On the other hand, road-side pedestrians can be used as a relay between two vehicles, thus filling this hole.

This thesis exploits the regularity of pedestrian movement and uses it to dissemi-

nate data between road-side pedestrians. To decrease the latency of communication, this dissertation explores the tradeoff between latency and redundancy, in light of the fact that smartphone carried by pedestrians have a limited power supply. It may be possible to deliver packets more quickly if more than one copy of a packet is redundantly scattered across the network. In the context of human users, the scattering strategy can be done intelligently to take advantage of social behaviors. Although this forces the redundancy to be a function of social clusters in the network, the corresponding reduction in latency can be worthwhile. Moreover, it may be feasible to operate at a desired point on the latency-redundancy tradeoff spectrum based on the requirements of the particular application.

In addition to the regularity of the pedestrian social behavior, we also note that their mobility includes a large proportion of *irregular* deviations from habitual patterns. Our studies of real mobility traces show that deviations are not rare, and, more importantly, they are difficult to parameterize through simple predictions. Thus, the worst-case performance of mobility-assisted forwarding is limited by these deviations, which often results in unacceptable performance. This motivates the need to design routing protocols that also cover irregular movements without incurring significant overhead.

During data dissemination, the authenticity of the data must be ensured, while the privacy of the vehicles should be preserved. As stated in the previous section, pseudonyms are used to protect vehicles' privacy, but they expose the system to Sybil attacks. Previous methods for preventing Sybil attacks depend on the hardware equipment on vehicles (directional antenna) or dense RSUs (short-term pseudonyms are distributed by RSUs one at a time instead of being preloaded by the CA all at

once).

The above issues motivate our research on a Sybil-attack protection scheme in which a vehicle is guaranteed to have a pseudonym to use at any time. The RSUs are only used to detect pseudonym abuse; therefore, the compromise of RSUs does not significantly impact the security or privacy of the whole system. Moreover, our scheme does not require hardware support such as directional antennae and attack detection only relies on the messages.

The proposed Sybil-attack detection scheme relies on RSUs. As a semi-trusted party, the compromise of an RSU has been considered to be an important issue in VANET security. However, most prior work is focused on error handling after the detection of a compromised RSU or on efforts that can reduce the impact of a compromised RSU. There is a need for a distributed and fast method to detect and revoke a compromised RSUs; this need motivates our work on the development of a scalable and low-cost solution to RSU compromise that can quickly detect and revoke compromised RSUs.

1.5 Thesis Outline

Chapter 2 proposes a GPS-free vehicle localization that works in a large area with short prediction delay. Only inexpensive built-in sensor readings are used for the location prediction. The error in the location prediction can be quickly determined and corrected using signals from cellular towers and nearby WiFi access points. Localization is carried out in two stages; the first is used for training, and the second is used for prediction. In the training stage, GPS signals, sensor readings, and cellular tower signals are used to train the system. In the prediction stage, the

system becomes independent from the GPS signals, and simply uses sensor readings and cellular tower signals to predict the vehicle's location.

Chapter 3 proposes a protocol to cope with irregular movements (thereby reducing the worst-case latency) by replicating the packets at carefully chosen node subsets, thus achieving a smaller worst-case delay. The main concept is to scatter multiple copies of a packet in the network such that even deviant nodes will be "close" to at least one of these packets. The number of copies must be suitably chosen to limit overhead. Additionally, the packets must be carefully routed such that the destination nodes (whether in habitual or deviant mobility) encounter the packet within a bounded time. Thus, with a given bound on the delivery delay, we can pre-select the relaying nodes and distribute the packets. Towards this end, our routing algorithm – Diverse Routing (DR) – statistically clusters the network into encounter-based social clusters. If the clustering can be done appropriately, even deviating nodes will be a part of a cluster at any given time. In this framework, if copies of a packet can be disseminated to at least one member of each cluster, the worst-case delivery delay can be reduced significantly. Because the number of clusters is far fewer than the total number of network nodes, the replication overhead is moderate. Moreover, this algorithm can be easily extended to multicast routing without introducing extra computation or communication overhead. DR builds on these observations and provides a scalable, delay-bounded routing protocol.

Chapter 4 presents the privacy-preserving Sybil-attack detection scheme. In this scheme, we assume that an RSU is securely connected to the DMV via a backhaul-wired network and that the DMV plays the role of the CA and has the ability to manage vehicle registration, ownership, and other administrative policies. Our

scheme requires the DMV to provide vehicles with a pool of pseudonyms that are used for hiding the vehicle's unique identity. To prevent a vehicle from using multiple pseudonyms for a Sybil attack, the pseudonyms assigned to a particular vehicle are hashed to a common value. By calculating the hashed values of the overheard pseudonyms, an RSU and the DMV will be able to determine whether the pseudonyms came from the same pool, thus helping to identify a Sybil attack. In this scheme, privacy is preserved as long as the RSU can be trusted. However, a compromised RSU may be able to "single out" a vehicle by assimilating all of the pseudonyms hashing to a unique value. Several other challenges arise while attempting to incorporate both privacy and security into a vehicular network system. We discuss these challenges and address them systematically through a light-weight, scalable protocol that we call "Privacy Preserving Detection of Abuses of Pseudonyms" (P^2DAP). The details of P^2DAP are presented in Chapter 4.2. Though our algorithm requires the calculation/storage of pseudonyms, we show that the computational and storage overhead are practical.

As a follow-up to P^2DAP, Chapter 5 describes a compromised-RSU detection and revocation scheme. In this scheme, an RSU must obtain its authentication key from m nearby nodes (where $m > 0$ is a predetermined design parameter) based on Shamir secret sharing (Shamir, 1979). A watchdog mechanism (Marti et al., 2000) is used such that a compromised RSU can be detected and disabled by nearby low-cost sensor nodes. Our scheme is named Prediction and Local-Collaboration-Based Authentication (PLCA). This solution increases the resiliency of the network to the compromise of an RSU and the subsequent injection of false data, but with low hardware cost and low communication overhead.

2

GPS-free Localization in a VANET

Vehicle localization is an important research topic for the implementation of VANETs. Although the use of GPS has been widely considered for this purpose, there are many geographical areas that are not covered by GPS, such as urban areas with lots of buildings (Melgard et al., 1994) and low-latitude areas during geomagnetic storms (Basu et al., 2001). To deal with this situation, researchers have proposed various methods to allow localization in these areas (Bo et al., 2013; Han et al., 2012). However, these solutions either attempt to cover short segments of road that GPS do not have a GPS signal or produce errors that increase with time.

In this chapter, we propose a GPS-free vehicle localization method for VANETs. As in (Bo et al., 2013; Han et al., 2012), we assume that each vehicle carries a smartphone for sensing and communicating. The smartphone's sensor readings are used for landmarks and dead reckoning, and its cellular signal is used to adjust the localization estimates. By combining both methods, we achieve GPS-free localization

over a large area with relatively smaller errors.

The sections in this chapter are organized as follows. Section 2.1 introduces the basic concept of our GPS-free localization algorithm. Section 2.2 provides the technical details for the GPS-free localization algorithm and discusses the two stages and the three components in the algorithm. Support Vector Machines (SVMs) and Hidden Markov Models (HMMs) are combined to estimate the road segment on which a vehicle is currently driving. Another SVM is used to calculate the distance that a vehicle has driven inside a given road segment. Lastly, Section 2.3 presents experimental results, and Section 2.4 presents conclusions.

2.1 Overview of a GPS-Free Localization System

2.1.1 System Assumptions

We made the following assumptions in our work.

1. First, we assume that the area has an intermittent GPS signal. In other words, there are times in which the GPS signal can be collected and there are times in which the GPS signal is poor and cannot be detected. (Shagimuratov et al., 2007) describes an area that satisfies such an assumption. In this scenario, we are able to train the system with GPS readings and use it to make location predictions when the GPS signal is weak.

2. Next, we assume that vehicles on the same road segment at the same time in the day have consistent behavior. This is not always true, but we can easily classify drivers based on their speed with a survey. Data collected by the drivers with similar speeds can be shared among multiple vehicles to help localization

on all of them.

3. Finally, we assume that a vehicle always knows its starting point. This is a reasonable assumption, considering that, when stopped, a vehicle driver has sufficient opportunity to determine the current location. On the other hand, checking the location of a moving vehicle is not feasible. Although map matching methods (Yang et al., 2003; Newson and Krumm, 2009) can be used to identify a vehicle trajectory, as researchers did in (Han et al., 2012), this method has a long prediction delay. In particular, the vehicle must travel for a sufficiently long distance (several kilometers) to ensure the accuracy of the map matching.

2.1.2 Basic Framework

The idea of a GPS-free localization system is inspired from speech-recognition (Shannon et al., 1995; Ganapathiraju et al., 2000), in which SVMs (Cortes and Vapnik, 1995) are used to recognize the single speech units (phonemes) and hidden Markov models (HMMs) (Rabiner and Juang, 1986) are used to represent the temporal dynamics of speech. We adopt such a system for path recognition. First, a path can be divided into multiple segments with various shapes, and SVMs are then used to recognize the shapes of each road segment. Next, after we obtain the shapes of all of the road segments on a path, HMMs are used to recognize the entire path.

Support Vector Machines

SVMs are linear classifiers that learn from a labeled dataset and then predict the labels of a different dataset, and they are widely used to analyze data and recognize patterns. Given a training set $\{(x_1, y_1), \cdots, (x_N, y_N)\}$—where each $x_i \in \Re^n$ is a

multi-dimensional datapoint and $y_i \in \{-1, 1\}$ is the label— a kernel function $k(\mathbf{a}, \mathbf{b})$, and a penalty parameter C, we can train the SVM by solving the following quadratic problem:

Find the vector α that maximizes

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

with constraints

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

and

$$0 \leqslant \alpha_i \leqslant C$$

Once α is obtained, the SVM predicts the label y of a new datapoint x by calculating

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i k(x_i, x)$$

Following this step, we have

$$y = \begin{cases} -1 & f(x) \leqslant 0 \\ 1 & \text{if} f(x) > 0 \end{cases}$$

The SVM described above is a binary SVM; however, most of the SVMs that we use in our work are multi-class SVMs. Multi-class SVMs can be constructed by combining multiple binary SVMs. To create a K-class SVM, we train K binary SVMs, each of which can recognize one class of data. This method of creating a multi-class

SVM is based on a one-versus-rest approach (Kreßel, 1999; Schölkopf and Smola, 2002).

Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical tool that is widely used in signal processing, especially speech processing. HMMs model generative sequences where the underlying interchangeable states generate an observable sequence. For the convenience of describing an HMM, we define $Q = \{q_1, \cdots, q_L\}$ as a sequence of states with length L and $O = \{o_1, \cdots, o_L\}$ as the corresponding observation sequence.

An HMM contains the following two assumptions:

1. Each state is dependent only on the previous state; i.e.,

$$P(q_i|q_1, \cdots, q_{i-1}) = P(q_i|q_{i-1})$$

2. Each observation is dependent only on the current state; i.e,

$$P(o_i|o_1, \cdots, o_{i-1}.q_1, \cdots, q_i) = P(o_i|q_i)$$

An HMM can be described by the following tuple:

$$\lambda = (\mathbf{S}, \mathbf{V}, \mathbf{\Pi}, \mathbf{\Theta}, \mathbf{\Phi})$$

where $\mathbf{S} = \{s_1, \cdots, s_N\}$ represents the finite set of states of the Markov model; $\mathbf{V} = \{v_1, cdots, v_T\}$ is the finite set of observations that are observable;

$$\mathbf{\Theta} = \begin{pmatrix} \theta_{1,1} & \theta_{1,2} & \cdots & \theta_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ \theta_{N,1} & \theta_{N,2} & \cdots & \theta_{N,N} \end{pmatrix}$$

is the $N \times N$ state transition matrix and $\theta_{m,n} = P(q_i = s_n | q_{i-1} = s_m)$ is the probability that state s_m transitions to state s_n;

$$\mathbf{\Phi} = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,T} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,T} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{N,1} & \phi_{N,2} & \cdots & \phi_{N,T} \end{pmatrix}$$

is the $N \times T$ emission matrix, in which $\phi_{m,n} = P(o_i = v_n | q_i = s_m)$ is the probability that o_n is observed at state s_m; and $\Pi = \{\pi_1, \cdots, \pi_n\}$ is the initial state distribution with $\pi_n = P(q_1 = n)$.

HMM Training

In a typical HMM training problem, we are given a sequence of observations O and must derive Π, Θ, and Φ from the sequence. Various algorithms can be used to solve this problem; example algorithms are described in (Baggenstoss, 2001; Collins, 2002). However, we will show in later sections that, in the context of this dissertation, both Q and O are known in the training stage, which greatly simplifies the HMM training. The parameters Π, Θ, and Φ can be easily determined from Q and O using the basic definitions described above.

HMM Decoding

In HMM decoding, we are given an HMM and a sequence of observations O and must derive the underlying state sequence Q such that

$$Q = \text{argmax}_Q P(O|Q)$$

This is equivalent to our objective of determining a vehicle's location from a sequence of observed road statuses.

The Viterbi algorithm (G.D., 1973) is commonly used as a dynamic-programming solution to decode an HMM. The algorithm can be described by the following pseudocode:

Algorithm 1 Viterbi Algorithm for HMM Decoding.

Require: $O = \{o_1, \cdots, o_L\}$, Π, Θ, and Φ
Ensure: $Q = \{q_1, \cdots, q_L\}$ such that PO—Q is maximized
  Define $L \times N$ matrices V and Ptr
  $V(1,1) \Leftarrow 1$
  $V(l,1) \Leftarrow 0$, for all $1 < l \leqslant L$
  for $i = 2$ to L **do**
    for $j = 1$ to N **do**
      $V(i,j) \Leftarrow \phi_{j,o_i} \times \max_k \theta_{k,j} V_k(i-1)$
      $Ptr(i,j) \Leftarrow \text{argmax}_k \theta_{k,j} V(i-1,k)$
  $q_L \Leftarrow \text{argmax}_k V(L,k)$
  for $i = L$ downto 2 **do**
    $q_{i-1} \Leftarrow Ptr(i, q_i)$

2.1.3 System Overview

The GPS-free localization procedure is divided into two stages: training and prediction. In the training stage, the vehicle collects all of the sensor readings and the GPS readings. Then, the data is used to train the following three types of predictors:

1. The SVM that predicts the shape of a given road segment (e.g., the orientation of the road, whether there is a stop/turn, etc.).

2. The HMM that predicts the location of a road segment in a given area.

3. The regression SVM that performs the dead reckoning inside a road segment.

Once the predictors are trained, the location of the vehicle can be obtained from the predictions. Figure 2.1 shows how the vehicle location is derived from the prediction results.

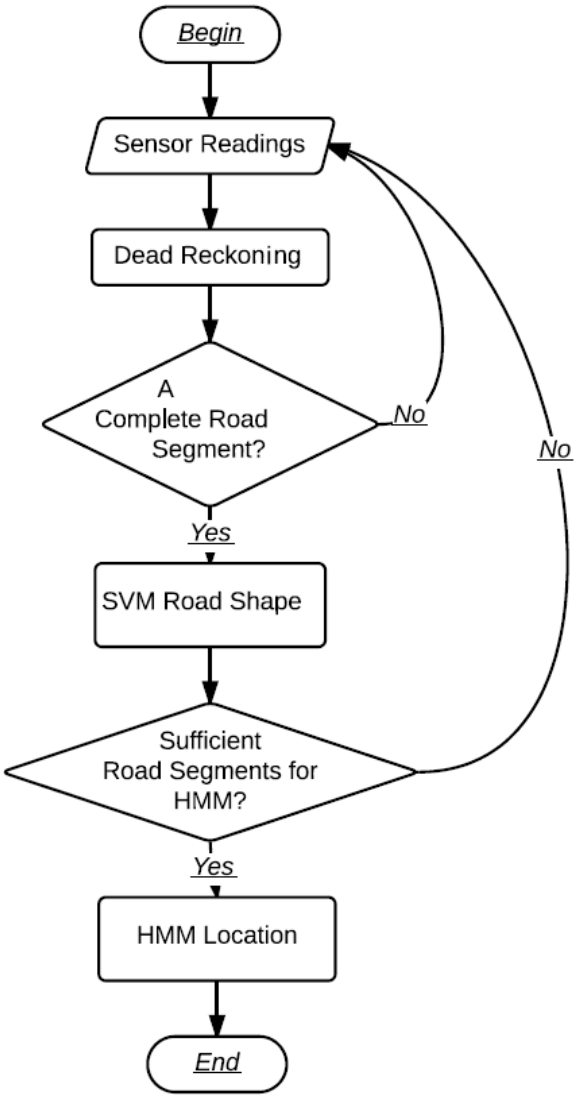


FIGURE 2.1: The flowchart for the GPS-free localization procedure.

Given a starting point and the dead reckoning prediction, the system can determine the following:

1. How long a vehicle has been on a particular road segment, and

2. When the road segment will end.

After ensuring that the vehicle has driven through a complete road segment, the shape of the road segment is derived from the collected sensor readings. Once this derivation is completed, the end point of the derived road segment is set as the starting point of the next road segment. The HMM is used when a vehicle encounters a road crossing, in which case we can determine what path the vehicle has taken. Finally, to prevent prediction errors from accumulating over time, WiFi signals and cellular signals (in Android phones, these signals are combined to determine a coarse-grained estimate of the smartphone's location) are used to correct the deviated location estimation.

2.2 GPS-Free Localization System

2.2.1 Road Shape Derivation

In this subsection, we describe the derivation of the road shape. In (Bhoraskar et al., 2012), (Bo et al., 2013), and (Ahmed et al., 2013), machine-learning techniques are used to derive the motion of a vehicle from smartphone readings. In our work, we use the same information to derive the shape of the road. The road shape contains the following information: direction, turns, and stops.

Direction of a Road

The most straightforward method to derive a vehicle's direction is to obtain it from the smartphone's compass reading, an approach that works well in certain road segments. Figure 2.2 shows the compass readings for three different road segments. As can be seen, the readings match the direction of the road quite well.

However, the compass reading from a smartphone is not always stable; the reading sometimes fluctuates. We address this problem by combining the gyroscope readings with the compass reading. The two sensors record the possible change of a smartphone's speed and direction and can be used to handle the fluctuations inherent in the compass reading, i.e., on seeing fluctuations in compass readings, the gyroscpoe readings can help to determine whether the fluctuation is caused by change in orientation or just false alarms. Figure 2.3 shows a compass reading with significant fluctuations and a low gyroscope reading that indicates the direction has not changed significantly. Therefore, we can combine the compass and gyroscope readings to determine whether a turn has actually occurred.

Sharp Turns

A "sharp turn" refers to a turn at a road crossing where, in most cases, vehicles are forced to stop or slow down before making the turn. Such actions leave obvious fingerprints in the accelerometer and gyroscope readings. Figures 2.4 – 2.5 show sensor readings for sharp turns to the left and right.

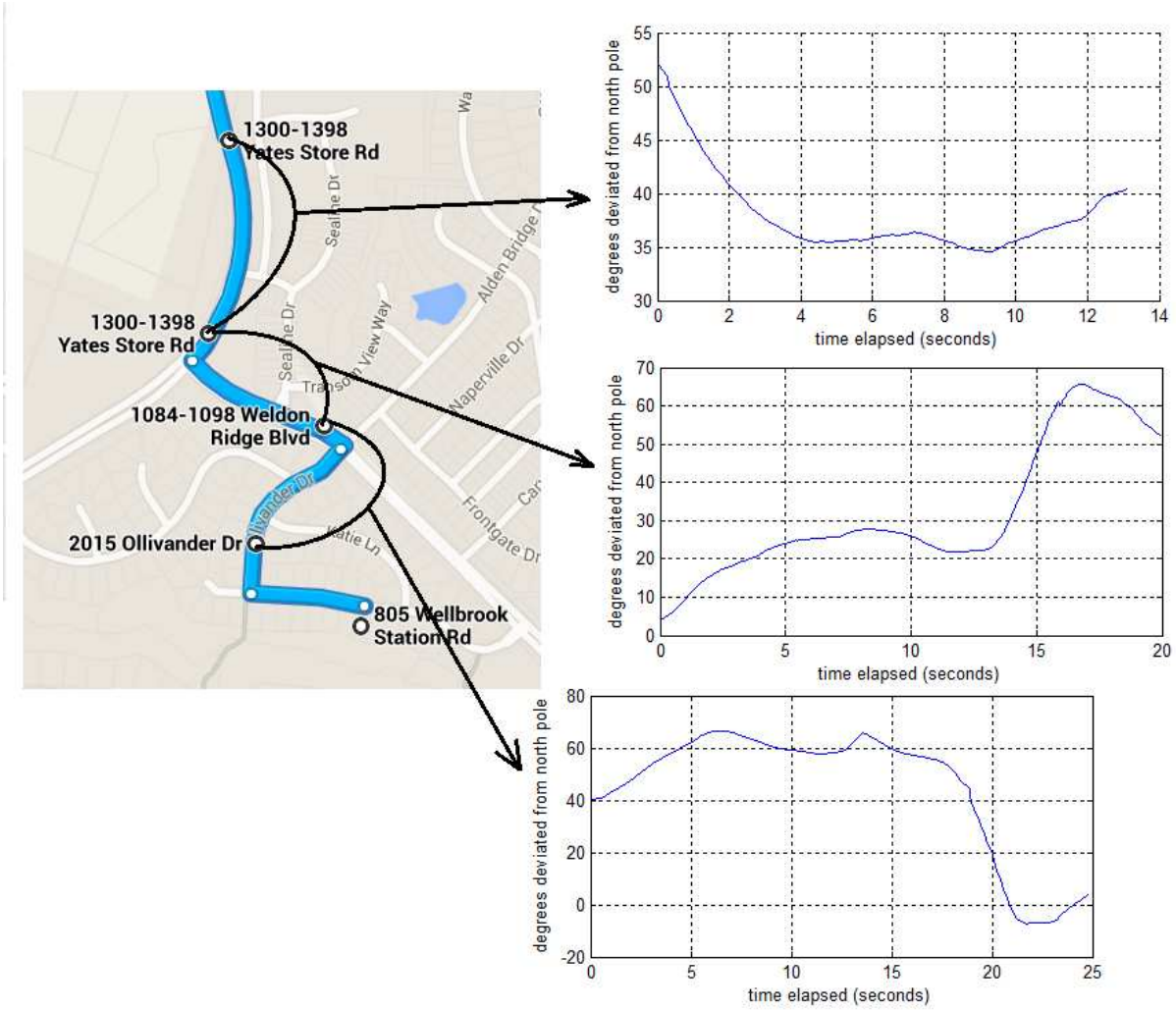


FIGURE 2.2: Compass readings for three different road segments.

Slight Turns

A "slight turn" is a smaller turn, such as one that might occur on a curvy road. Compared to the sensor readings from a straight road, the readings from a curvy road show more changes. The blue (dotted) path in Figure 2.6 shows a road segment

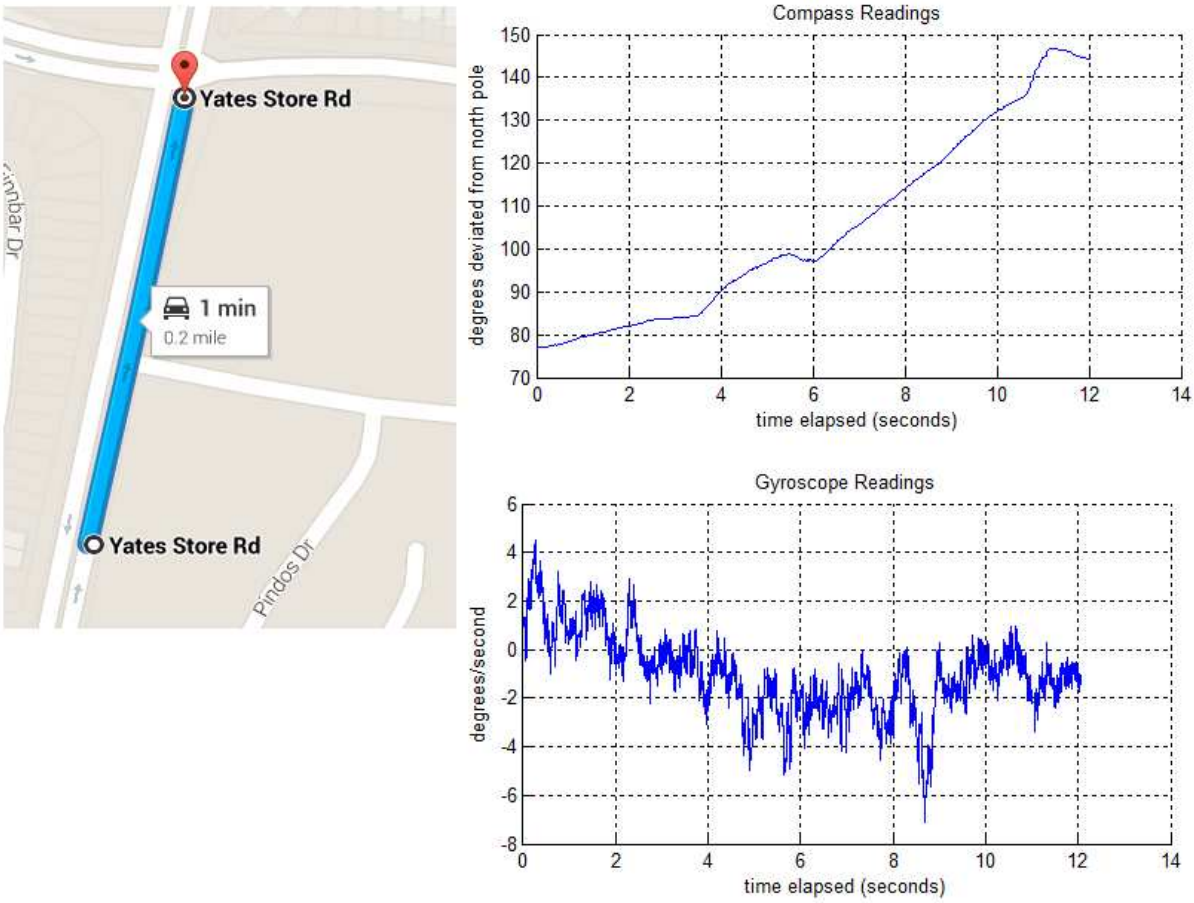
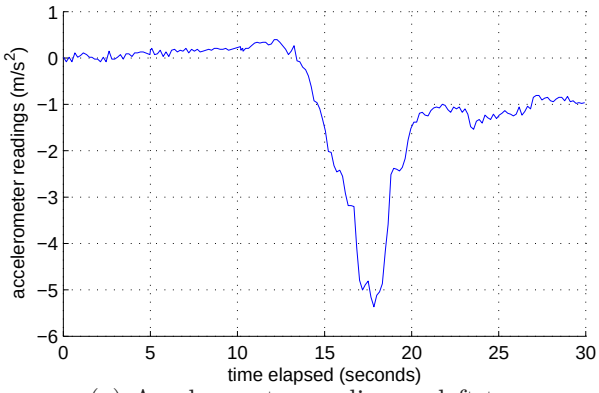


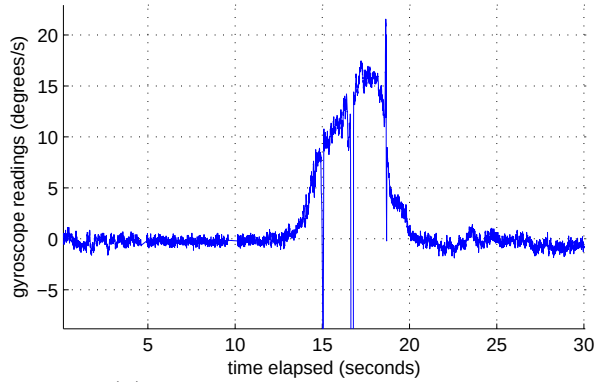
FIGURE 2.3: Comparison between compass readings and gyroscope readings: compass readings show fluctuations but gyroscope readings show no turns.

with a slight right turn.

However, when compared to the readings for a sharp turn, a slight turn lacks the distinctive signature of the vehicle slowing down/stopping. Comparisons between the three different road segments (straight, sharp turns, and slight turns) are shown in Figure 2.7. As seen in the figure, a slight turn generates some change the in accelerometer readings, but they are not as obvious as those generated by a sharp



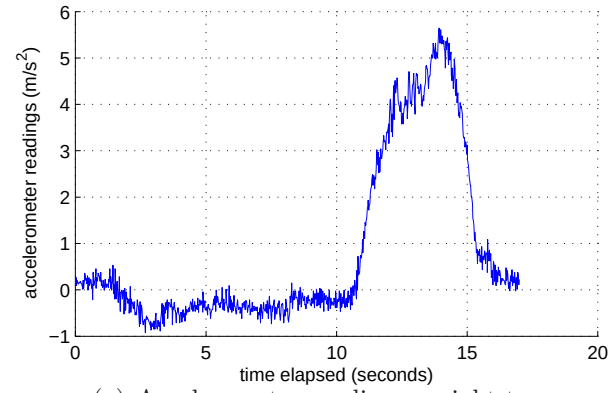
(a) Accelerometer reading on left turn



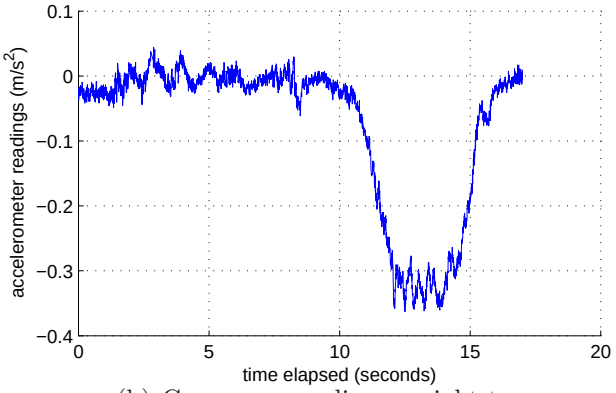
(b) Gyroscope reading on left turn

FIGURE 2.4: Accelerometer and gyroscope readings during a sharp left turn.

turn. The gyroscope readings on a slight-turn road segment show the same trend as the accelerometer readings. Among the sensor readings from the three different kinds of road segments, the readings from the straight road are closest to zero, as expected. The readings from a slight-turn road are closer to zero than those from a sharp-turn road but farther from zero than those from a straight road.



(a) Accelerometer reading on right turn



(b) Gyroscope reading on right turn

FIGURE 2.5: Accelerometer and gyroscope readings during a sharp right turn.

Stops

Stops usually indicate a traffic light or a stop sign. Similar to sharp turns, stops also have distinctive signatures in the accelerometer and gyroscope readings; specifically, the sensor readings remain unchanged for a relatively long time (seconds to minutes). In Figure 2.9(a) and Figure 2.9(b), we can see the signatures of a long stop followed by a left turn.

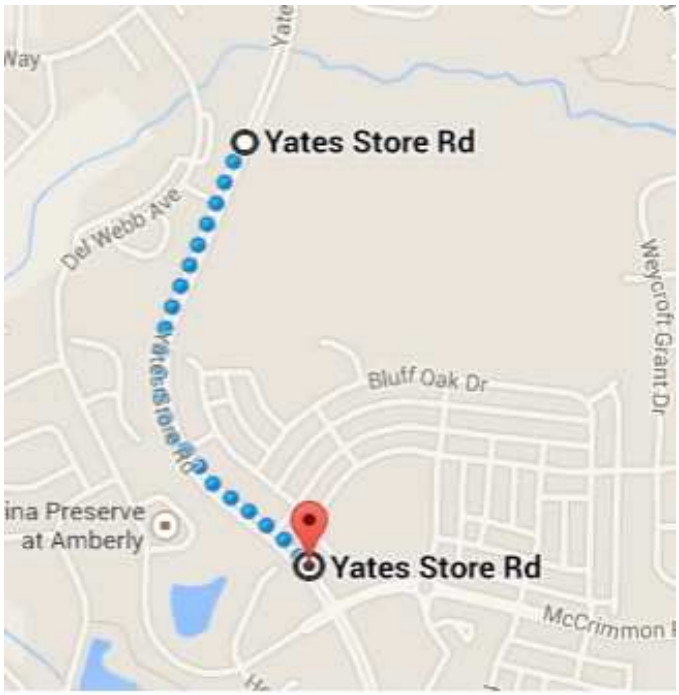


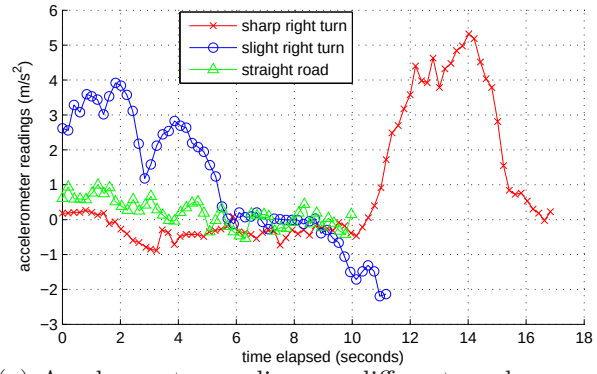
FIGURE 2.6: A slight right turn on a map.

Feature Extraction, Selection, and SVM Training

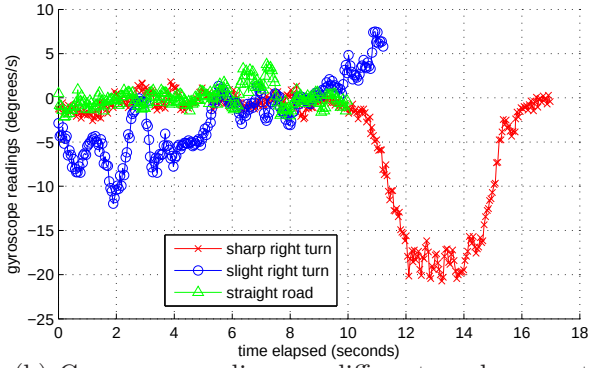
The following features are extracted from sensor readings for every road segment:

- Accelerometer readings: minimum value, maximum value, correlation between readings along different axes, zero-crossing times, and integral over time.
- Compass readings: mean value.
- Gyroscope readings: minimum value, maximum value, correlation between readings along different axis, zero-crossing times, and integral over time

The RELIEF feature-selection algorithm from (Kira and Rendell, 1992) is then used to select features that would best train an SVM. Note that RELIEF cannot recognize



(a) Accelerometer readings on different road segments



(b) Gyroscope readings on different road segments

FIGURE 2.7: Comparison between sensor readings on sharp turn, slight turn, and straight road segments.

overlapped features, which results in some redundancy in the selected features, but it is very robust against errors. In this scenario, feature redundancy is not a major concern because an SVM can be trained on an offline server and loaded onto the smartphone at a later time. Because the smartphone sensor readings tend to contain errors, the RELIEF algorithm is an appropriate choice for this application.

After extracting the features, four different SVMs are trained, one each to detect sharp turns, curvy roads, stops, and the road direction. The prediction results of the

SVMs are then combined into a vector to form the observation of the road segment.

Estimating the Border of a Road Segment

When training the SVMs, we can simply use the GPS readings to tell which road segment the vehicle is in. However, when using the SVMs to predict the shape of a road segment, we assume no GPS readings are available. Therefore, only the sensor readings and cellular/WiFi signals can be used to estimate whether the vehicle has driven through a complete road segment. We use a simple method to detect the border of a road segment, described as follows.

When driving on a road, a vehicle continuously extracts the features for the sensor readings that are collected, and compares the features to those in the training dataset by calculating their difference from the features for training. We name such a difference as "feature difference". When given the starting point of a road segment, which can be obtained from earlier SVM/HMM predictions, the vehicle knows the road segments on which the training data are collected. It is expected that the feature difference starts at a large value, and decreases as the vehicle approaches the other end of the road segment. After the vehicle passes by the border of this road segment, the difference should start to increase.

Figure 2.8 shows a decrease in feature difference as the vehicle drives along a road segment. We see that the feature difference quickly drops and reaches the minimum value when the vehicle is on the other end of the road segment. After that, the readings slight fluctuates. From the figure, it can be seen that it is feasible to use the sensor readings' features to estimate whether a vehicle as reached the end of a road segment.

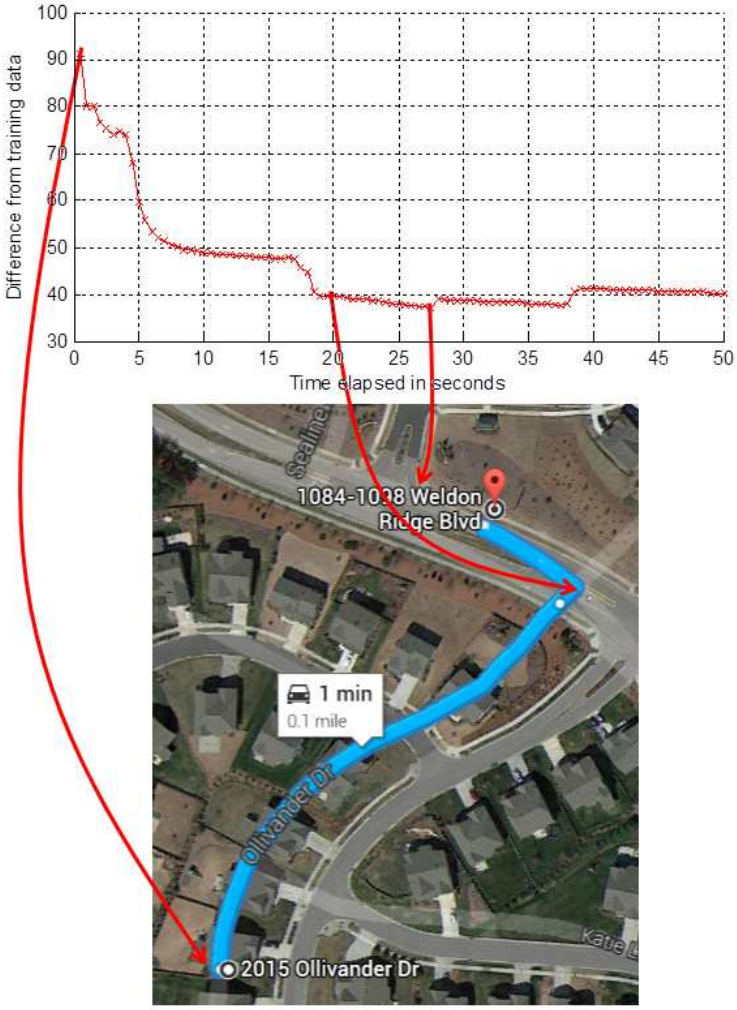


FIGURE 2.8: Calculated feature difference at different points on a road segment.

2.2.2 Collaborative Sensing

In the real world, there can be scenarios in which multiple vehicles close to each other are driving on the same road in the same direction and maintain contact with one another through a wireless network. These clustered vehicles can be exploited for higher accuracy in the road shape prediction and vehicle localization. Although a prediction from data sensed by a single vehicle can have errors, we expect the prediction from data collected by multiple vehicles to have a reduced error. One straightforward method to perform collaborative sensing is to allow vehicles that are close to one another to vote on the road shape. Each vehicle independently predicts the road shape from its own sensed data and then broadcasts it to its one-hop neighbors. On receiving the predicted road shapes from the nearby vehicles, a vehicle then uses the road shape that most neighbors agree with as the current road shape. We will show in Section 2.3.7 that collaboration between even 3 vehicles can greatly improve the accuracy of the predictions.

There are cases in which vehicles close to one another have different road shape predictions. For example, vehicles driving on opposite directions on the same road can have different prediction on the road direction. Also, we can encounter an even more complex scenario at road crossings, where vehicles can predict various road shapes such as left turns, right turns, or a straight road. In this dissertation, we filter out these cases and only consider predictions from nearby vehicles that have relatively longer contact.

2.2.3 Dead Reckoning inside a Road Segment

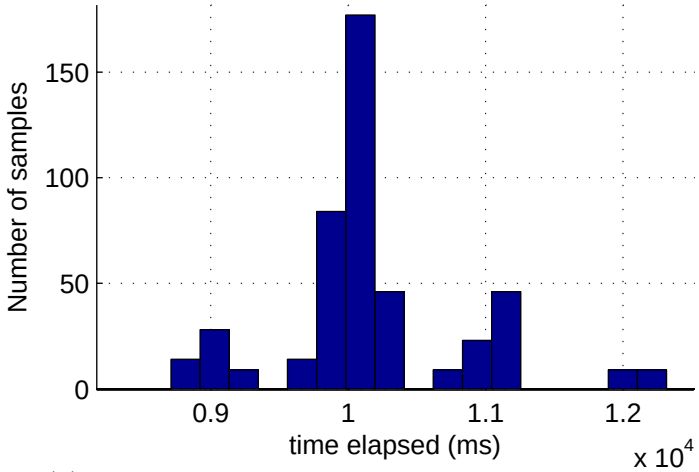
We use dead reckoning inside each road segment to further improve the location prediction accuracy. Different dead reckoning methods are used with differently shaped road segments based on the vehicle's behavior. Figure 2.9 explains the different road segment types and Figure 2.10 presents a comparison of the time it takes a vehicle to drive along two different road segments of similar length.

Note that a vehicle takes 8.5 – 12.5 seconds to pass through road segment 1, which is straight (i.e., no stops/turns). Road segment 2 has a left turn at the stop sign, and therefore a vehicle takes 18 – 33 seconds to go through, which has a much larger variation in speeds. We can immediately see that it is straightforward to estimate the distance a vehicle has driven in road segment 1 using the time elapsed.

We now focus on the dead reckoning inside road segment 2. Because vehicles usually stop or turn in such a road segment, we expect the sensor readings to dramatically change if a stop/turn occurs. This can help us to estimate where we are inside the road segment. To help clarify the sensor reading inside such a road segment, we divide road segment 2 into ten different subsegments. Figure 2.11 contains the cumulative accelerometer readings along the x axis when a vehicle is driven through different subsegments. Readings collected from two different devices on multiple vehicle runs are displayed for each subsegment to show the shape of sensor readings.

We can see that the shape of the sensor readings in the same subsegments are similar for all runs and devices. Therefore, the sensor reading can assist us in dead reckoning inside a given road segment. We reuse the extracted features in Section

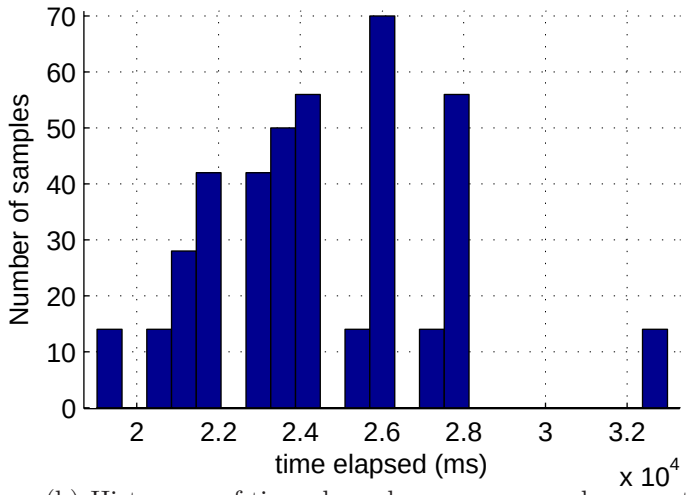
(a) Straight road segment



(b) Histogram of time elapsed on a straight road segment

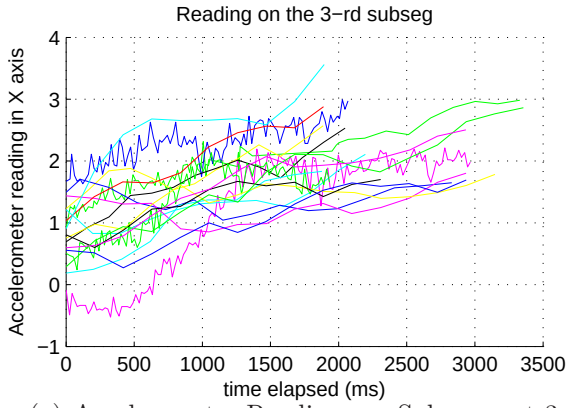
FIGURE 2.9: Time elapsed on a straight road segment.

(a) Curvy road segment

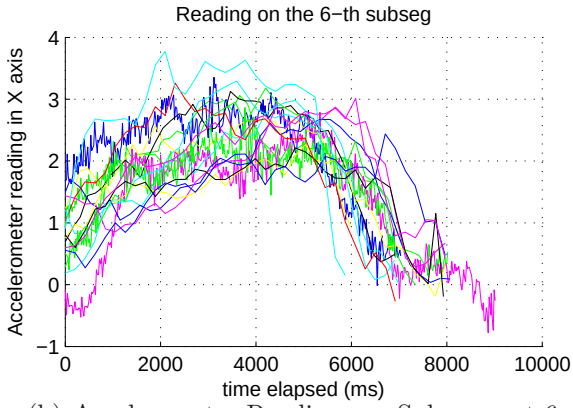


(b) Histogram of time elapsed on a curvy road segment

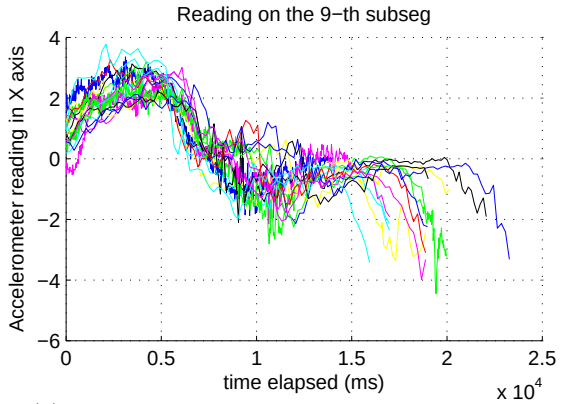
FIGURE 2.10: Time elapsed on a curvy road segment.



(a) Accelerometer Readings on Subsegment 3



(b) Accelerometer Readings on Subsegment 6



(c) Accelerometer Readings on Subsegment 9

FIGURE 2.11: Comparison of cumulative accelerometer reading in different subsegments.

2.2.1 as well as two additional features (time elapsed and the double integration of accelerometer readings) to perform the dead reckoning. Again, the RELIEF feature-selection algorithm is used to pick the features that are later used to train the SVM and make predictions. Each road segment has an individual SVM. However, instead of extracting the features over a road segment, the features are extracted every second from the cumulative sensor readings. All of the features are then used to train an epsilon-SVR, which is used to predict the location of a vehicle inside a road segment.

This yields a finer-grained estimate for the vehicle location than if only the road segments were used.

2.2.4 HMM for Localizing a Road Segment

With a sequence of observed road segments, we can use HMM decoding to derive the current location of the road segment. Considering that a path can contain many road segments, we divide paths into a set of subpaths that each contain six to seven road segments. This granularity can be easily achieved using the built-in coarse-grained location readings from the smartphone, which are calculated from cellular tower signals. Then, we apply both HMM training, which can be performed with the existing data, and HMM decoding to every subpath. The derived road shape in Section 2.2.1 is used as the observation sequence O in the HMM, and the actual location of the road segment is the state sequence Q.

When given a starting point, we can derive the hidden state of the other road segments from the observation. When the vehicle encounters a road crossing, our results show that the smartphone tends to produce an error of up to two road segments due to the error in observations. However, this error can always be corrected with the coarse-grained location derived from the cellular tower and WiFi signals, thus not exceeding the granularity of the coarse-grained location.

2.3 Experiments

In this section, we describe our experiments and the results.

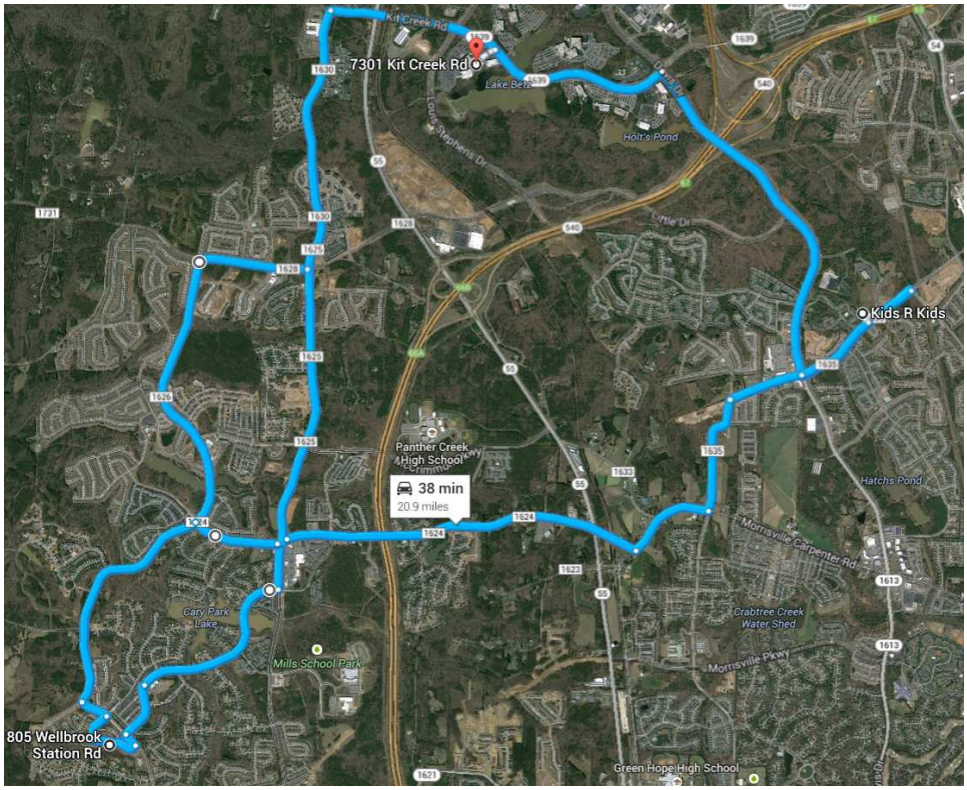


FIGURE 2.12: The paths on which we collect the data for training and prediction.

2.3.1 Experimental Environment

Two Samsung Nexus S phones with Android 2.2.3 and one Samsung Nexus phone with Android 4.1.0 are used for data collection and analysis. All data are collected at the same time of day (during the morning around 9 am and the evening around 6 pm) to avoid the impact of traffic changes at different times. The data are taken in the Research Triangle Park (RTP) area of North Carolina, along three one-way paths shown in Figure 2.12.

Each drive generates a dataset, and a total number of 108 datasets are recorded

along the paths. Twenty datasets are used for training data, and the remainder (88 datasets) are used for testing and evaluation. The process of location prediction is simulated on Matlab.

2.3.2 Expanding the Training Dataset

Because the training dataset is not large enough to adequately train an SVM, we expand the training set by carefully extending the sensor readings. The sensor readings are extended in the following two ways:

1. Temporal extension. The sensor readings can vary based on the vehicle's speed and the length of waiting time in front of a road crossing. To simulate this difference, we extended the readings using two methods:

   (a) Split the readings into different segments, and stretch or shrink the readings based on the simulated speed.

   (b) At a road crossing where a vehicle must stop (i.e., a stop sign or traffic light), add or delete waiting time.

2. Spatial extension. The sensor readings can vary due to external noise and slight bumps on the road. To simulate these deviations, the noise in the sensor readings is collected via a high-pass filter and then added to the simulated sensor readings.

With the above extensions, we eventually generated 600 datasets for training. Of these training datasets, 300 are used to train the SVM that predicts the road segment shapes and 300 are used to train the HMM that predicts the road segment locations.

2.3.3 Road Segment Recognition

The sensor readings in testing dataset are divided into per-segment readings using the method described in Section 2.2.1. The mean road segment length is 266.72 meters for all datasets. The standard deviation of the road segment lengths is 60.67 meters, while the standard deviation of the road segment lengths in the training dataset is only 22.24 meters. The relatively larger fluctuation in the calculated road segment lengths in the testing dataset is expected, because the road segments in the training dataset are calculated from GPS readings, which can provide a much more accurate value of distance between two waypoints than the other sensors.

Most of the fluctuations in road segment lengths happen on several consecutive road segments on a straight road. On such a road, the mean difference of road segment lengths between testing dataset and training data set is 85.04 meters. On the other hand, on road segments near a turn or stop, such a difference only has mean value of 30.80 meters, which is a very small value comparing to the 266.72 meters' mean road segment length. Therefore, the fluctuation of the road segment lengths mostly happens on consecutive straight road segments. Such a fluctuation almost cannot affect the prediction of the road shape, because the length of a straight road segment length has little impact on its predicted shape.

2.3.4 Road-Shape Prediction

SVMs are used to distinguish stops, left turns, right turns, and the four directions (east, west, south, and north). The accuracy of the predictions is shown in Table 2.1. Note that we have false-positive and false-negative percentages only for the SVM that predicts stops and non-stops. This is because all of the the remaining SVMs

Table 2.1: Prediction accuracy of road shapes as percentages.

Road Shape	Stop	Sharp Turns	Slight Turns	Direction
Accuracy	95.94	92.11	85.75	56.46
False Positive	3.09	N/A	N/A	N/A
False Negative	7.41	N/A	N/A	N/A

are multi-class SVMs that contain 3 or more classes, and thus we can only calculate the prediction accuracy and not the false flag percentages.

In recent related work (Bhoraskar et al., 2012), the false-positive and false-negative percentages for detecting vehicle breaks are 2.7% and 21.6% respectively. Therefore, compared to prior work, the false positives are comparable but the false negatives are significantly reduced. One concern is that the prediction accuracy for the road direction is only 56.46%. However, we will soon show that this problem can be overcome by using the observed road shape to train and decode HMMs. Also, in the scenario of multiple vehicles, we have a great opportunity to improve this accuracy.

2.3.5 Dead Reckoning

In our experiments, each road segment has a length of approximately 267 meters. To achieve better granularity in predicting a vehicle's location, we divide each road segment into 10 subsegments and then extract the features each time the vehicle reaches each segment. The dead reckoning inside all road segments has a mean error of 64.97 meters and a standard deviation of 41.78 meters.

FIGURE 2.13: Two paths splitting.

2.3.6 HMM Prediction

The HMMs are trained with the predicted road shapes on a set of consecutive road segments. After obtaining the road shape for a complete road segment, we can now use the HMM to derive where a road segment is on the path. On a path without branches, this is a relatively easy task, and HMM decoding can achieve a 100% accuracy. We focus on the road segments where the vehicle can take different paths, as shown in Figure 2.13.

Sixty datasets are selected based on these two different paths, thirty for path 1 and thirty for path 2. Table 2.2 shows the HMM decoding results for the two paths.

These results have a prediction accuracy of 71.7%. For the incorrect predictions,

Table 2.2: Prediction accuracy of HMM for the crossing of path 1 and path 2.

Actual Path	Predicted as Path 1	Predicted as Path 2
Path 1	20	10
Path 2	23	7

we can always make a correction using the next available coarse-grained reading. As shown in 2.2, the errors are quickly corrected because the next coarse-grained readings are obtained soon after the vehicle passes the crossing. During this time period, a maximum error of 457.87 meters is introduced. This error is generated after an incorrect HMM prediction is made immediately before being corrected by the coarse-grained location information.

2.3.7 Prediction with Multiple Vehicles

Due to the limited number of vehicles used in the experiment, we simulate the scenario of multiple vehicles driving together by taking readings on the same road segment that were collected multiple times. We then randomly pick 1, 3, or 5 reading samples (depending on the theoretical number of multiple vehicles), make an SVM prediction on each of the samples, and then take the road shape predicted by the majority of vehicles as the collaborative prediction of multiple vehicles. We run such a simulation 100 times. Table 2.3 compares the prediction accuracy for different numbers of vehicles.

We observe a dramatic increase in the accuracy of the prediction of the road direction, and other predictions also show slight improvement. The results show that collaborative sensing can greatly improve the prediction accuracy of the road

Table 2.3: Prediction accuracy of road shapes as percentages with multiple vehicles.

Number of Vehicles	Stop	Sharp Turns	Slight Turns	Direction
1	95.94	92.11	85.75	56.46
3	98.26	93.11	86.06	83.55
5	98.74	94.58	86.48	84.16
7	99.17	94.74	86.81	84.75

Table 2.4: Overall prediction accuracy for all paths.

Algorithm	Mean Error (m)	Std Error (m)	Max Error(m)
GPS-free prediction (single vehicle)	67.40	51.84	457.87
GPS-free prediction (three vehicles)	64.68	42.68	245.42
Coarse-grained location	510.7	498.4	2536.24

shape. Additionally, we notice that the improvement is only slight when the number of vehicles is greater than three.

Next, we use the collaboratively-sensed road segment shapes as the observations in the HMM and derive the location. We use the same dataset as in Section 2.3.6, i.e., thirty readings on path 1 and thirty readings on path 2. This time, we observe 100% accuracy in the HMM prediction that identifies path 1 or path 2. However, out of the sixty test cases, two test cases have slight prediction errors that mistakenly predict a road segment on path 1 as its successive road segment. We then conclude that, in the case of collaborative sensing, the localization error is mostly introduced by the errors in the dead reckoning and the subsequent road shape prediction.

2.3.8 Quantification of Overall Errors

We now combine all observations to derive the predicted location of a vehicle. Table 2.4 shows the comparison of GPS-free localization with the built-in coarse-grained location readings in Android. From the table, we see that with the aggregation of sensor readings, the error in localization without GPS is greatly reduced.

2.4 Conclusions

We have described a localization system that predicts a vehicle's location without GPS readings. The effectiveness of this system has been tested in RTP, North Carolina. From the results, we can see that when the sensor readings and cellular tower signals are combined, we can greatly improve the accuracy of localization. This algorithm is very useful in regions where the GPS signal is unstable or is often disturbed or blocked.

3

Diverse Routing: Data Dissemination in Ad Hoc Networks

In this chapter, we explain the concept of diverse routing. Our main idea stems from observing the mobility patterns of real users in several university campuses. In addition to the clustering property and regularity of movements, we observed two major properties of the movements of nodes:

1. When deviating from its usual pattern, a node is likely to be reached by nodes in the different social clusters instead of being isolated, and

2. The number and size of social clusters depend on the parameter chosen to determine the social relationship among the nodes.

From the above study, we recognized the benefit of partitioning users into virtual groups, such that members of each group have pre-determined opportunities for mutual communication. We then distribute a copy of the packet to each virtual

group, such that at any time, because the destination is likely to reside in any one of the virtual groups, both the average and worst case latency of communication can be low. The proposed diverse routing protocol is developed from this idea. Our main contributions are listed as follows.

1. Analysis of realistic mobility traces and characterization of opportunities for routing. We assimilated daily activity patterns from anonymous students, faculty, and staff at Duke University, and used them to construct mobility traces. We studied mobility patterns from three different traces, namely, the Duke campus mobility survey, UCSD Wireless Topology Discovery (WTD) data set, and MIT Reality Mining (MITRM) data set. We characterized behavioral patterns, with a focus on deviations. We showed that deviations are not rare and can seriously impact the end-to-end latency in delay tolerant network (DTN) routing. We designed models and metrics to analyze the traces and gain useful insights.

2. Draw upon behavioral insights to design a routing protocol that copes with deviations from habitual mobility and exploits these nodes for quality solutions. We designed a routing algorithm that can deliver a packet with limited delay to the destination, even if the latter deviates from habitual activity and even if the intermediate nodes or the destination deviates from regular movement patterns. We show that this goal can be achieved with only a moderate increase in overhead and provide a control to adjust the tradeoff between delivery delay and overhead. We show that the delivery delay and overhead are tunable and that it is feasible to achieve latencies that are only slightly different from

optimal (i.e., the delivery delay achieved by epidemic routing) with moderate redundancy.

3. Performance evaluation through comparison with the optimal and with existing routing protocols. We simulated our DR protocol using the ns2 simulator and compared it against two other popular protocols – Epidemic and Simbet (Vahdat and Becker, 2000; Daly and Haahr, 2007). Our experimental results show that we can approach the shortest delay with a bounded difference at any time while incurring a moderate packet redundancy.

3.1 Mobility Measurements and Metrics

In recent years, many efforts have been invested in developing mobility models for social activity patterns (Tuduce and Gross, 2005; Kim et al., 2006; Chaintreau et al., 2006; Rhee et al., 2011; Musolesi et al., 2004). These models can be divided into two categories – trace-based (Tuduce and Gross, 2005; Kim et al., 2006; Chaintreau et al., 2006; Rhee et al., 2011) and theoretical (Musolesi et al., 2004). Trace-based models are bottom-up in the sense that the exact mobility paths are known and they are suitably aggregated into a parameterized model. While these models are realistic, they lack the ability of generalization. In other words, a model derived from one trace cannot be expected to accurately describe another trace. On the other hand, theoretical models are top-down. They model aggregate mobility patterns through a set of parameters and instantiate this model to produce individual mobility paths. Although theoretical models offer the ability to simulate large-scale mobility across large numbers of nodes, they may not accurately reflect realistic patterns.

Developing the above mobility models is not the focus of this chapter; instead, the DR algorithm proposed here is based on node activities in real traces. A similar study has been done in (Hui et al., 2008), in which researchers present the social clusters formed in real-life DTN scenarios, and propose a routing protocol based on the social clusters. An important observation in (Hui et al., 2008) is that different social clusters can be formed by defining different metrics of social relationships. In our work, we carefully examine this behavior and exploit it for routing purposes.

This thesis studies three real-life traces: the Duke trace, the MITRM trace (Eagle and Pentland, 2005), and the WTD trace (Cheng, 2008).

1. For the Duke trace, we conducted a survey on 54 volunteering users. Volunteers anonymously filled out the survey, stating their timings and movements on a typical day. We then used the survey results to generate an approximate trace for each user.

2. In the MITRM trace, 100 smart phones were distributed to people on the MIT campus, and these phones maintained records of phone calls, cell towers, and Bluetooth communications for 9 months. In this chapter, we only take their Bluetooth activities in one month for our study.

3. The WTD trace maintains logs of wireless LAN connections between mobile devices and access points (APs), and there are a total of 275 mobile hosts recorded for two and a half months.

3.1.1 Extracted Features of the Traces

In order to develop a routing algorithm that can limit the delivery delay, we investigate features related to the communication latency. The latency between two nodes is affected by the nodes' frequency of encounter and the length of the node encounters. Therefore, we considered the use of the following features to describe the latency between nodes.

Normalized Contact Duration ($\Delta_{i,j}$)

The normalized duration that two nodes are within each other's communication range is defined as follows:

$$\Delta_{i,j} \equiv \frac{\int_0^T \delta_{i,j}(t)dt}{T}, \tag{3.1}$$

where T is the time elapsed and $\delta_{i,j}(t)$ is the length of time for which nodes n_i and n_j encounter one another. In other words,

$$\delta_{i,j}(t) = \begin{cases} 1 & \text{if } n_i \text{ and } n_j \text{are in mutual contact,} \\ 0 & \text{if } n_i \text{ and } n_j \text{are not in contact,} \end{cases} \tag{3.2}$$

This quantity has been examined in (Hui et al., 2008) and used as a metric to evaluate the social proximity between nodes.

Inter-Encounter Time ($\Theta_{i,j}$)

One problem of $\Delta_{i,j}$ is that it may not be able to distinguish node pairs with different frequencies of encounters. For example, consider two cases shown in Figure 3.1. Observe that both node pairs have the same value of $\Delta_{i,j}$. However, the movement

in Figure 3.1(b) is more effective for opportunistic communication. This is because periodic separation and union enable each node to "fetch" data from other parts of the network and deliver them to each other. Therefore, we need to study another feature that can capture the inter-encounter time $\Theta_{i,j}$ between two nodes n_i and n_j, defined as follows.

$$\Theta_{i,j} \equiv \frac{\int_0^T \theta_{i,j}(t)dt}{T} \tag{3.3}$$

where $\theta_{i,j}(t)$ is the length of time, starting from time t, until the time when n_i and n_j encounter each other.

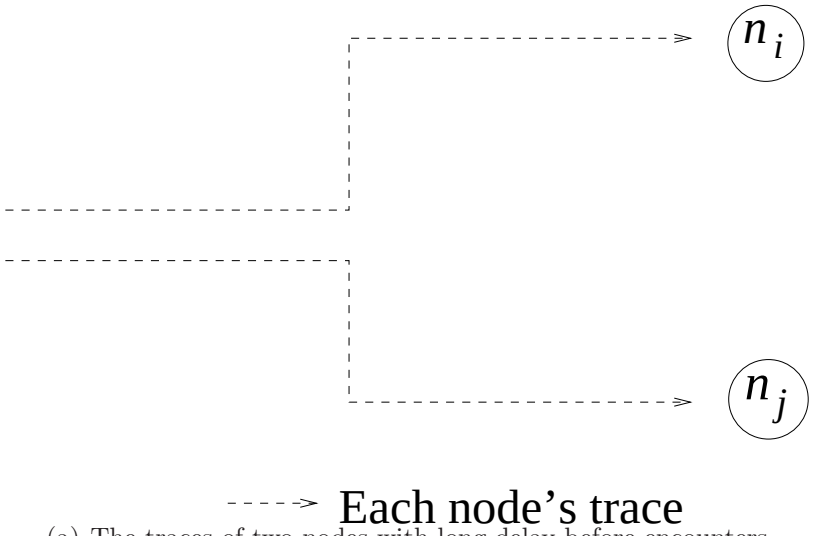
$$\theta_{i,j}(t) = \min(t')|_{t' \geqslant t, \delta_{i,j}(t')=1} - t \tag{3.4}$$

Note that $\Theta_{i,i} = 0$ for all values of i. We observe that $\Theta_{i,j}$ is the average time interval between encounters between nodes n_i and n_j in the past. Hence, a smaller value of $\Theta_{i,j}$ indicates a larger frequency of encounters between nodes n_i and n_j.

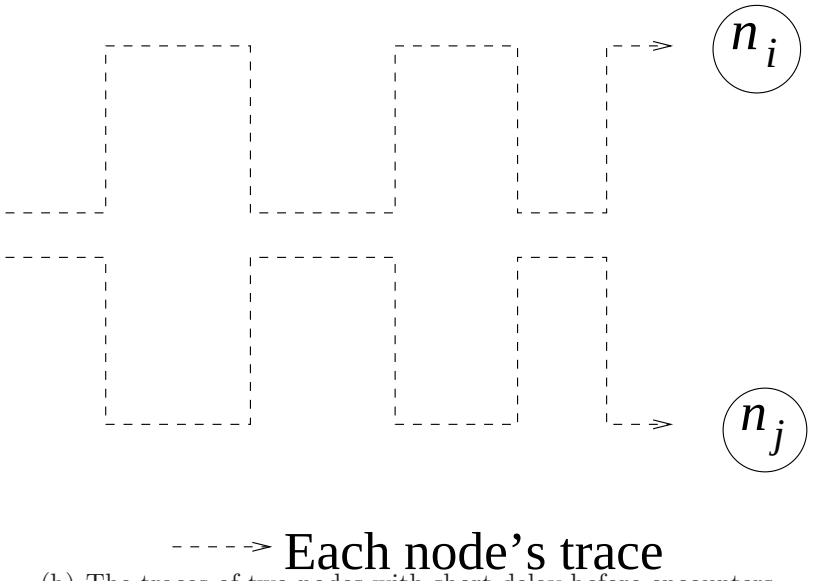
Because we focus on campus activities for all of the traces, we only examine the activities during working hours, i.e., from 9:00 am to 5:00 pm every day for all three traces.

3.2 Mobility Characterization

This section characterizes human mobility patterns from real traces and identifies opportunities for handling the nodes that deviate from their normal behaviors. Specially, we study the behavior of nodes deviating from habitual clusters. Before explaining our study, we first define the concept of social neighbor and clusters as follows.



(a) The traces of two nodes with long delay before encounters



(b) The traces of two nodes with short delay before encounters

FIGURE 3.1: The traces of node pairs with the same $\Delta_{i,j}$.

Definition 3.1. *Two nodes n_i and n_j are considered to be social neighbors if and only if $\Theta_{i,j} \leqslant \tau$, where τ is a pre-defined threshold of the average inter-encounter interval.*

This definition is similar to the connection-threshold-based social relationship in (Musolesi et al., 2004). With this definition, we can construct a *network graph* G for a DTN, in which each vertex represents a node and an edge is formed between two nodes if and only if they are social neighbors. Following the definition of social clusters in (Palla et al., 2007), we define *social clusters* in DTN as k-cliques in G. Partitioning the DTN into social clusters then becomes a k-clique covering problem; i.e., the objective is to find the k-cliques that can cover G, where a k-clique is an undirected graph in which each vertex has a degree greater than k. We now begin with the characterization of node activities and then use these insights to explain the intuition behind our protocol.

3.2.1 Irregularity of Movements

The irregularity of movements indicates that a node n_0 can occasionally leave its social neighbors. If we give a packet to n_0's neighbor n_1 and expect n_1 to pass the packet to n_0 soon, the irregular movements of n_0 will cause unexpectedly long delivery delay. To address this issue, we first find out where n_0 goes when it deviates from its social neighbors. If n_0 is simply isolated from all the other nodes during this time period, our only choice is to wait until n_0 revisits its social neighbors. However, if n_0 is visiting other non-neighbor nodes when it is away from neighbors, there are opportunities for improving the current social-cluster-based routing algorithms.

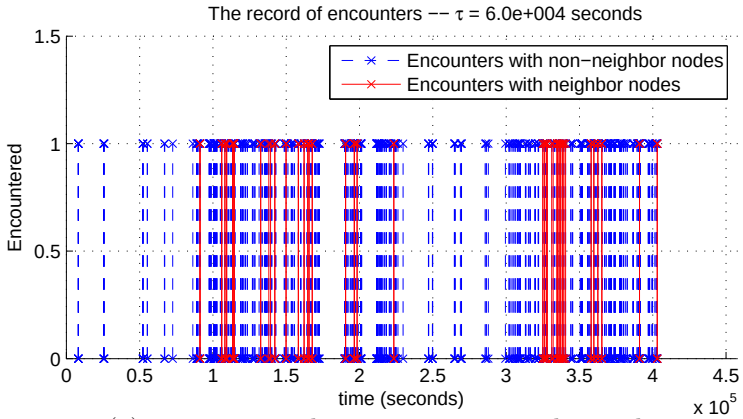
Figure 3.2(a) shows all of the encounters at n_0 in the MITRM trace. In this figure,

we see that during the periods that n_0 has a long delay before encountering any of its social neighbors, it regularly encounters non-neighbor nodes. Figure 3.2(b) shows a statistical summary of the encounters between n_0 and all the other nodes. In this figure, the x axis represents the Θ value between n_0 and the nodes it encountered and the y axis represents the number of encounters for that Θ value. From this figure, we observe that among all the node encounters of n_0, approximately 14% of the encounters are with nodes whose Θ values are greater than 4.5×10^5 seconds. This observation means that the opportunity of encountering strangers is non-trivial. We have similar observations from the other two traces, but they are omitted here.

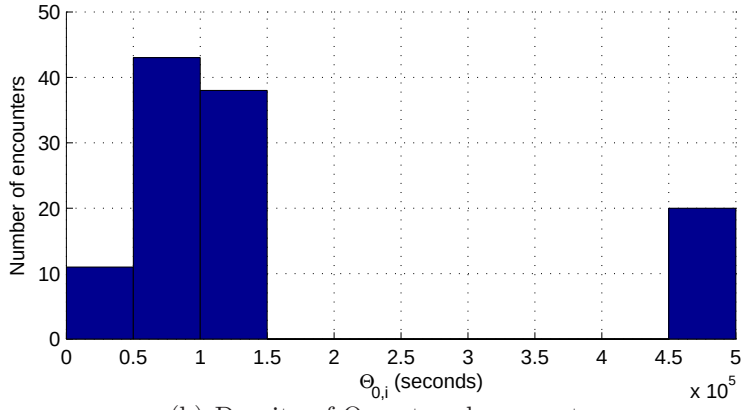
To obtain a global view of the nodes' activity outside of their social clusters, we first define a time-varying relationship between two nodes n_i and n_j, namely *social cover*.

Definition 3.2. *n_i and n_j are socially covered by each other at time t if and only if $\theta_{i,j}(t) \leqslant \tau$.*

When n_i is socially covered by n_j and its neighbors at time t, we consider n_i to be in n_j's social cluster at t. Note that social cover is different from social neighborhood. The former describes a temporary relationship, which varies quickly as nodes move. The latter is an average of the nodes' relationship over a long period of time, which changes slowly. We then define a metric Γ to evaluate the opportunity of exploiting a node n_i's non-neighbor nodes to deliver the packets to n_i. The measure $\Gamma(n_i)$ represents the proportion of time that n_i is socially covered by only non-neighbor



(a) Timepoints when n_0 encounters other nodes



(b) Density of $\Theta_{0,i}$ at each encounter

FIGURE 3.2: An overview of the activities of a node in the MITRM trace.

nodes, defined as

$$\Gamma(n_i) \equiv 1 - \frac{\int_{\theta_{i,j}(t)\leqslant\tau, j\in\text{neighbor}(n_i)} dt}{\int_{\theta_{i,j}(t)\leqslant\tau, j\neq i} dt} \tag{3.5}$$

A large value of $\Gamma(n_i)$ indicates that n_i often encounters strangers, thus being easily reached by non-neighbor nodes. We calculate the average value of $\Gamma(n_i)$, written as Γ, and the standard deviation of $\Gamma(n_i)$ for all the nodes in DTN with

Table 3.1: The Γ value with different thresholds for the three traces.

DataSet	τ (sec)	Γ	std($\Gamma(n_i)$)
Duke	1.8×10^3	0.4744	0.4366
Duke	3.6×10^3	0.1591	0.1110
Duke	5.4×10^3	0.0382	0.1503
WTD	3×10^4	0.7940	0.3641
WTD	6×10^4	0.5338	0.4633
WTD	9×10^4	0.3654	0.4390
WTD	1.2×10^5	0.1965	0.3573
MITRM	3×10^4	0.3462	0.2832
MITRM	6×10^4	0.3521	0.3137
MITRM	9×10^4	0.3062	0.3285
MITRM	1.2×10^5	0.2850	0.3176

various values of τ and for all of the on-campus traces. The results are listed in Table 3.1.

For all three traces, we observe that Γ values range between 0.03 and 0.8, which means that nodes are socially covered by strangers for a significant amount of time. From this observation, we draw two important conclusions:

1. Nodes can frequently leave their social neighbors and encounter strangers.

2. We have the opportunity to exploit encounters with strangers to deliver the packets.

From the above study, we see the necessity of coping with the irregular behavior of nodes during packet delivery. To achieve this goal, we must use the non-neighbor nodes as relaying nodes. These non-neighbor nodes should be chosen such that the destination can be reached by one of them with limited delay regardless of the destination location. To obtain an idea of which nodes to choose, we study the social

clusters formed in our traces.

3.2.2 k-Clique Cluster Detection

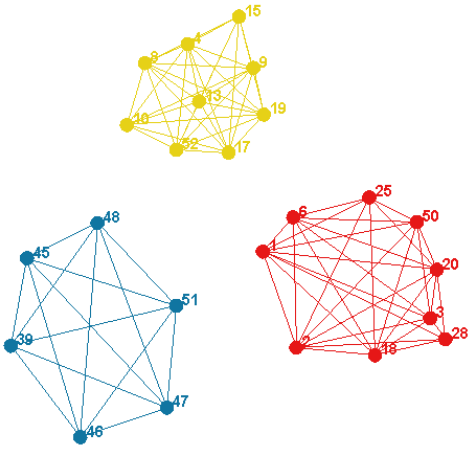
Similar to (Hui et al., 2008), we obtain the clusters of nodes using the k-clique cluster detection algorithm proposed in (Palla et al., 2007). We then construct the network graph G for each scenario with a threshold value of τ. Using this graph, we determine the clusters of nodes using the k-clique algorithm.

Figure 3.3 shows the k-clique social clusters detected in the Duke trace for τ =30, 60, and 90 minutes and $k = 5$, where all of the isolated nodes have been omitted. We see that the number of social clusters gradually decreases and the size of the social cluster increases as the value of τ increases. When $\tau = 90$ minutes, almost all of the nodes are in the same cluster with only a few isolated nodes. This observation shows that the social clusters can be adjusted by tuning the value of τ. Social clusters with smaller sizes can be obtained by defining a higher threshold of social relationship.

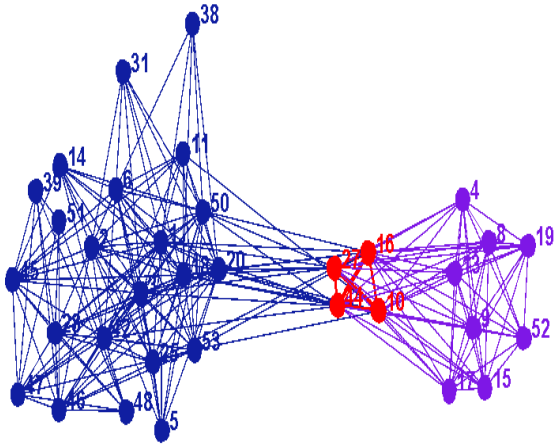
We performed the same analysis on the WTD and MITRM traces and observed similar trends. Figure 3.4 shows the k-clique social clusters in the MITRM traces for $\tau = 6 \times 10^4$, 9×10^4, and 12×10^4 seconds and $k = 4$. Again, we can observe the combination of social clusters as τ increases.

In Figure 3.4, note that when τ grows from 6×10^4 seconds to 9×10^4 seconds, the number of social clusters increases. This difference is because there are four isolated nodes when $\tau = 6 \times 10^4$ seconds, but they cluster together when $\tau = 9 \times 10^4$ seconds.

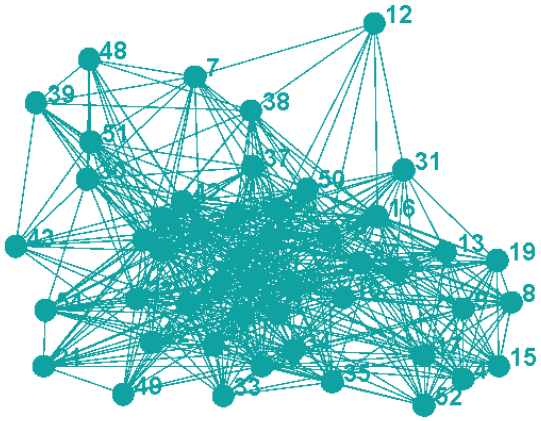
Figure 3.5 shows the k-clique social clusters in the UCSD WTD traces for $\tau = 3 \times 10^4$, 4.5×10^4, and 6×10^4 seconds and $k = 3$. To prevent the graph from being too congested, we only draw the social relationship of 100 nodes from the entire trace.



(a) $\tau = 1.8 \times 10^3$s

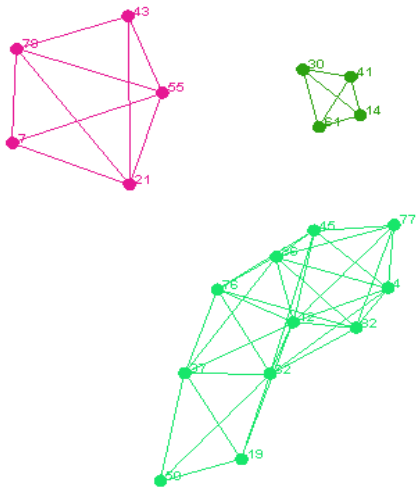


(b) $\tau = 3.6 \times 10^3$s

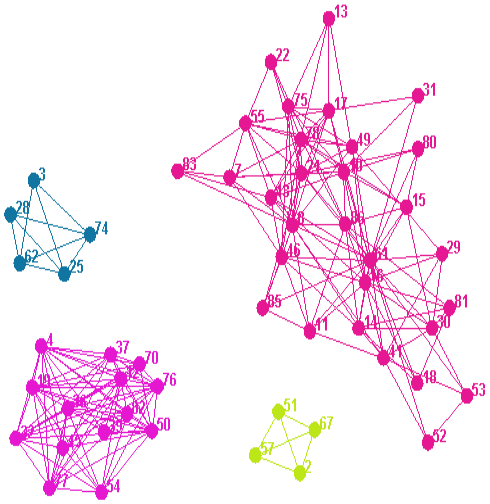


(c) $\tau = 5.4 \times 10^3$s

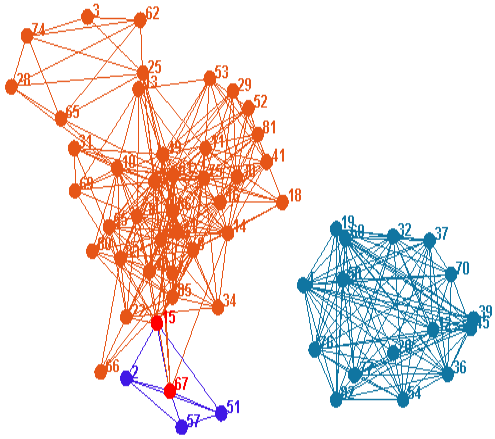
FIGURE 3.3: The k-clique social clusters formed in the Duke trace.



(a) $\tau = 6 \times 10^4$s

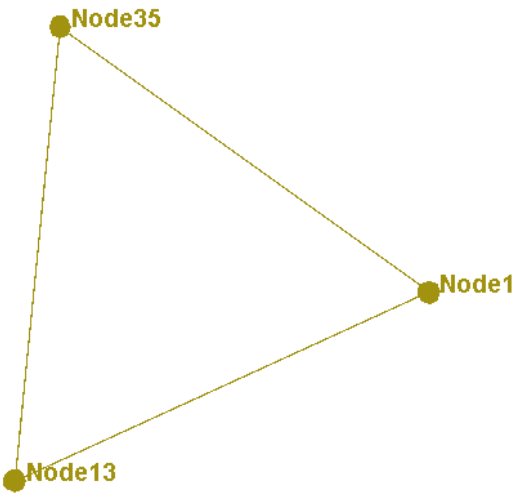


(b) $\tau = 9 \times 10^4$s

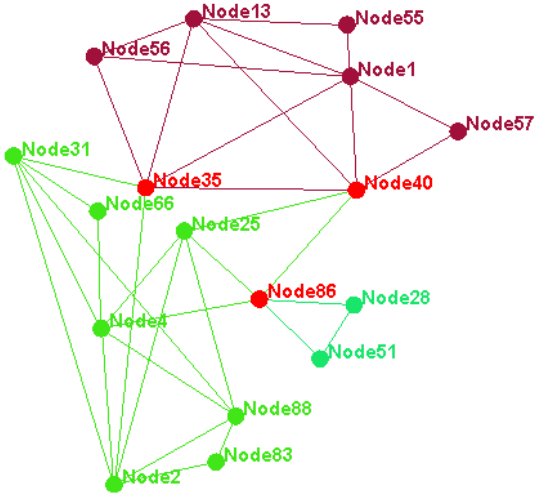


(c) $\tau = 12 \times 10^4$s

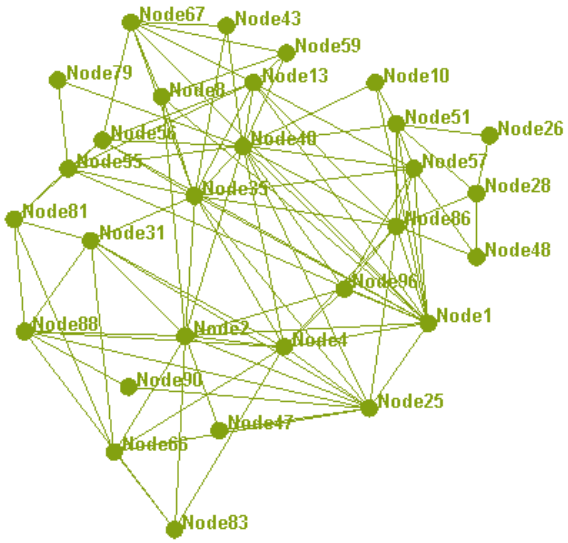
FIGURE 3.4: The k-clique social clusters formed in the MIT reality mining trace.



(a) $\tau = 3 \times 10^4$s



(b) $\tau = 4.5 \times 10^4$s



(c) $\tau = 6 \times 10^4$s

FIGURE 3.5: The k-clique social clusters formed in the UCSD WTD trace.

Note that, in contrast to the Duke and MIT traces, social clusters are not as obvious in the UCSD WTD trace. Instead, a small cluster is initially formed, and as the value of τ increases, indicating a looser threshold, the size of the cluster gradually grows.

An important feature of all three traces is that the level of social relationships can differ in terms of the inter-encounter time. Therefore, we can exploit different levels of social relationships to meet different requirements of the delivery delay.

This observation motivates the basic idea of our routing algorithm. First, we construct all possible social clusters such that the inter-encounter time between nodes in the same cluster is limited by a pre-defined delay threshold. Next, when sending a packet P, a copy of P is given to each cluster. Once P is distributed to all of the clusters, it can reach the destination within an adjustable delay. Because partitioning the DTN into social clusters is an NP-hard problem (Ding et al., 2001), we need a heuristic algorithm to identify to which relay nodes we should distribute P. More details are discussed in Section 3.3.

3.3 Protocol Design

In this section, we describe our DR protocol in more detail. The goal of our design is to (a) limit the delay of packet transmission within each social cluster, and (b) minimize the delay of packet transmission between different social clusters. With DR, we expect to achieve a delivery delay that is close to the optimum value. Our basic idea contains two steps. The first step is to divide the nodes into social clusters such that all of the nodes in the same social cluster have limited inter-encounter time; and the second step is to distribute the packets to all of the possible clusters.

DR is composed of three major components: (1) a clustering metric that formalizes the notion of pairwise user proximity, (2) a graph theoretic framework for spreading packets to multiple clusters, and (3) a protocol execution module that coordinates inter-cluster and intra-cluster communication.

3.3.1 Clustering Metric

In Section 3.2.2, we use $\Theta_{i,j}$, the average encounter interval between two nodes n_i and n_j, as the metric of the social neighbor between them. We then define a threshold τ and define two nodes as social neighbors if and only if $\Theta_{i,j} \leqslant \tau$. This measure is used as our clustering metric to define social neighbors.

3.3.2 Routing Framework

To route information from a source to a destination, we develop a graph-theoretic framework using social neighbors. For this purpose, we first construct a network graph G for the DTN, which is the same as in Section 3.2. The basic idea is to multicast each packet to a minimum subset of all of the nodes in DTN, $\mathcal{S}_\mathcal{D}$, such that any node in DTN is either in $\mathcal{S}_\mathcal{D}$ or is a neighbor of $\mathcal{S}_\mathcal{D}$. In other words, $\mathcal{S}_\mathcal{D}$ is the dominating set of G; thus, spreading the packet copies to the entire $\mathcal{S}_\mathcal{D}$ will result in a constrained delivery delay to the destination. Of course, members of $\mathcal{S}_\mathcal{D}$ may not be connected in G; hence, a connected dominating set $\mathcal{S}'_\mathcal{D}$ is required in order to efficiently propagate the packets. Because the computation of a connected dominating set is an NP-hard problem, a standard approximation algorithm (Guha and Khuller, 1998) is used in DR. The routing framework is described as follows.

Each source node constructs the graph G using the mechanism of periodic information exchange (described in Section 3.3.3). Then, the source node locally computes the connected dominating set (CDS) of G and disseminates a packet P to all members of the CDS (and the destination, n_D). Each member of the CDS opportunistically spreads the packet to other CDS members, or to the destination if possible. Because the destination is likely to be located near at least one social neighbor, the packet is expected to reach the destination quickly. Defining the social neighbor judiciously can result in low latency at the expense of reasonably low redundancy.

3.3.3 Protocol Description

A decentralized implementation of the routing framework requires two components: (1) an algorithm to compute the connected dominating set (CDS), and (2) a protocol to route copies of the packet to all of the members of CDS and, in turn, to the destination. We describe these components in this section.

(i) CDS Computation

Computing the CDS at a source node requires the knowledge of all of the social neighbors in the network. For this purpose, the source must gather the values of $\Theta_{i,j}$ for all node pairs. We have two options for calculating $\Theta_{i,j}$, depending on the equipment of the nodes.

Option 1 – Location-Based Algorithm

In this option, the nodes are assumed to be equipped with GPS. Each node periodically broadcasts a hello packet, piggybacked with the location vectors for each node. A location vector of node n_j consists of the past K locations of n_j

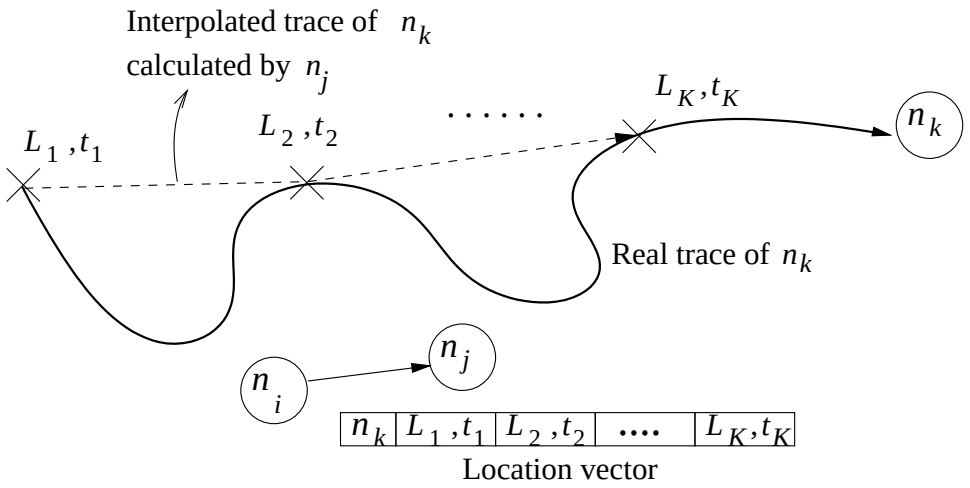


FIGURE 3.6: Estimating mobility traces from overheard location vectors.

at time-points $\{t_{curr} - nt_{step}\}_{n=0,1,2,...,K}$. If the location of a node is missing at a particular time-point, the location from a preceding time-point is used in its place. These vectors are cached at each node and are continually updated with more recent information. Figure 3.6 shows an example of this operation.

Once the location caches have warmed up, each node is expected to have location vectors for almost all of the other nodes. Now, if a source node n_s intends to send a data packet, it computes the CDS as follows: Firstly, the source node linearly interpolates the movements of each node and uses them to compute pairwise $\Theta_{i,j}$. Then, by thresholding $\Theta_{i,j}$ with τ, the source node constructs the social neighborhood graph. A heuristic algorithm is employed to obtain a good approximation of the CDS. This CDS is included in the header of the data packet and is multicast to social neighborhoods in the network. The routing component implements the multicast operation.

Option 2 – Location-Free Algorithm

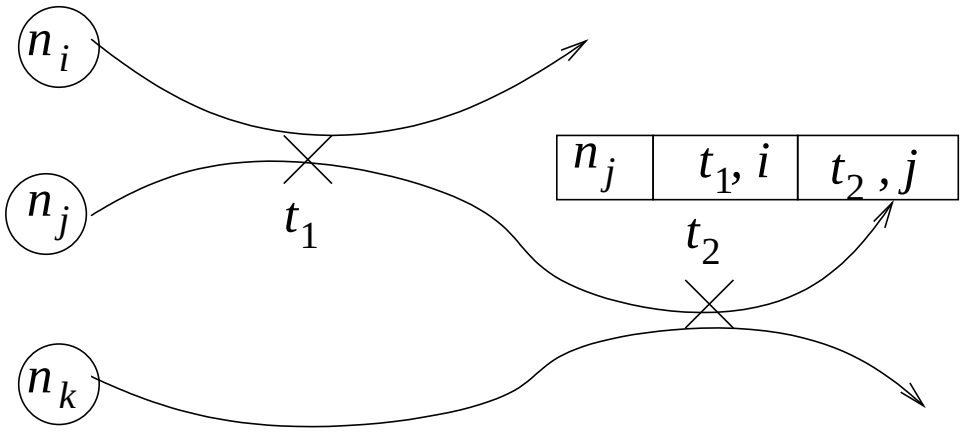


FIGURE 3.7: Generating location vector without GPS.

In this option, the nodes are assumed to have no location information. Initially, each node simply broadcasts a hello packet periodically. After a sufficient warm-up time, a node will accumulate a record of node IDs that it encounters. With this node ID record, we make a simple adjustment to the location-based algorithm presented above. In the construction of the node ID record, instead of snapshooting the coordinates of the nodes, a node records the smallest node ID that it encounters at each timepoint. For example, at time t_0, if node n_j encounters n_i and n_k, where $i < j < k$, then n_j will record the value $< t_0, i >$. If n_j is isolated at time t_1, then it simply records $< t_1, j >$. Figure 3.7 shows how a node ID record is constructed. In this scenario, node n_j encounters n_i at time t_1 and encounters n_k at time t_2. n_j records i and j at time points t_1 and t_2, respectively. This information is then used to generate the node ID record.

When interpolating the traces of the nodes, each node is assumed to remain at its location until the next recorded timepoint. After that, for each timepoint, two

nodes are considered to have encountered one another if and only if the node IDs in their node ID vectors at that timepoint are the same. The exact same algorithm is then used to calculate $\Theta_{i,j}$, construct the network graph, and calculate the CDS.

(ii) Packet Routing

By design, the members of the CDS are expected to encounter each other frequently. When a CDS member encounters another member, the former sends a copy of the packet to the latter. The packet spreads until it reaches all of the members of the CDS. Now, each CDS member consults its location cache to determine the most recent known coordinate of the destination. If a CDS member finds that the destination recently visited its cluster, it geographically forwards the packet towards the destination (Li et al., 2006). Geographic forwarding is expected to perform well here because the nodes in the same cluster are expected to be geographically close to each other. In a unicast scenario, once the destination receives the packet, it broadcasts an acknowledgment to the network. All copies of the packet within the DTN are then dropped. In a multicast scenario, the packets are stored in the buffers of relaying nodes and discarded in the FIFO manner when the buffer is full. However, packets with long paths (i.e., if the source node and destination node are far away from each other) are likely to be discarded from the FIFO packet queues before it is delivered. To solve this issue, an alternative method is to prioritize the packets and always discard the packets with the lowest priority, as proposed by many researchers in their DTN routing algorithms (Burgess et al., 2006; Ramanathan et al., 2007). We can see that, in a multicast scenario, the CDS is the same as in a unicast scenario. Therefore, when extending from the unicast to the multicast scenario, DR is not expected to add computation or communication overhead. In this dissertation,

we only discuss the unicast scenario and leave the multicast protocol as future work.

Unless carefully designed, the periodic distribution of location vectors or node ID records in DR can increase the control overhead. Although this overhead is amortized over all packets and DTN applications, we propose two optimizations to reduce it. First, these vectors are sent only after receiving a hello packet. Second, only the information of nodes that have undergone changes in the recent past are propagated. With the above improvements, the overhead in DR has a complexity of $O(m)$, where m is the average number of nodes whose information is updated in a node within a period of time[1]. We can further reduce m by tuning the length of the above time period. When taking the time period as 1 day, we have $m = 33$ for the MITRM trace and $m = 50$ for the WTD trace. This will yield approximately 100 – 500 bytes for each hello packet, which is a small overhead when broadcast with a proper interval. In our experiment, we have verified that this control overhead will not noticeably impact the performance.

3.4 Experimental Results

In this section, we use ns-2 to simulate DR and compare its performance to two other routing protocols – SimBet (Daly and Haahr, 2007) and epidemic routing (Vahdat and Becker, 2000). The basic idea of SimBet is to continue to forward each packet to both "bridge" nodes (i.e., the nodes that connect two or more social clusters) and the usual cluster of its destination until the packet is delivered. Simbet determines the "bridge" nodes by calculating the betweenness centrality of

[1] Note that if a node n_i have no news about n_j for a sufficiently long time, n_j is considered to be unchanged and thus is not included in n_i's hello packets.

each node and choosing the nodes with the largest betweenness centrality. Other social-relation-based DTN routing algorithms use a similar metric to determine the relaying nodes (Gao et al., 2009). SimBet only keeps one copy of each packet; thus, it is expected to have low communication overhead. On the other hand, epidemic routing floods each packet over the entire network such that the packet can be delivered with the shortest delay. Thus, the delivery delay of epidemic routing is considered to be the optimal delay. For each of the above three protocols, we examine the performance within the overall case and the worst case. In the overall case, we overview the delivery delay and communication for all of the packets. In the worst case, we examine the packets that are transmitted when the destination node or the relaying nodes happen to deviate from their usual cluster. In order to theoretically determine the worst cases, we need to (1) find all of the possible social clusters and relaying nodes, and (2) identify the timepoints when the bridge nodes and the destination node are not in their usual clusters. However, these two steps are non-trivial. In our simulation, we assume that the worst-case packets are those packets whose SimBet delay and epidemic delay have the largest difference in each test case. The reason for this assumption is as follows. A large difference between epidemic routing and SimBet routing indicates that relaying nodes chosen based on regular movements fail to deliver the packets quickly.

We run experiments on two on-campus traces the Duke trace and the WTD trace. With both traces, we show that the DR algorithm can be tuned to achieve a constrained delay.

3.4.1 Simulation Setup

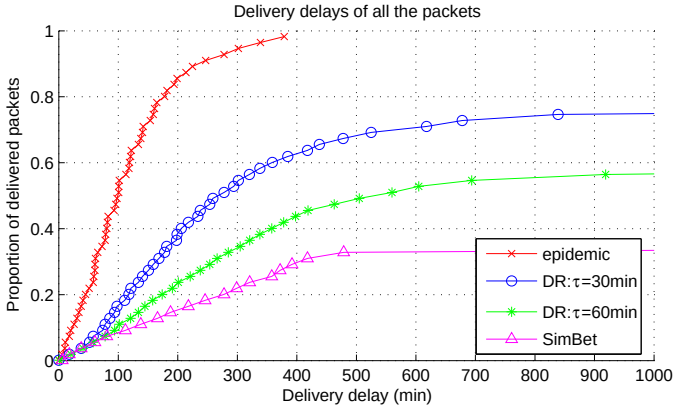
As mentioned in Section 3.1.1, we only used the trace during working hours for each day. We run 100 test cases for each experiment, and we randomly select a subset of nodes for each test case in our simulation. The size of each data packet is 1kB. We also assume that the communication range of each node is 80 meters and that the size of the data packet is 1kB. Table 3.2 shows the other parameters we used in our experiments.

Table 3.2: NS-2 simulation parameters.

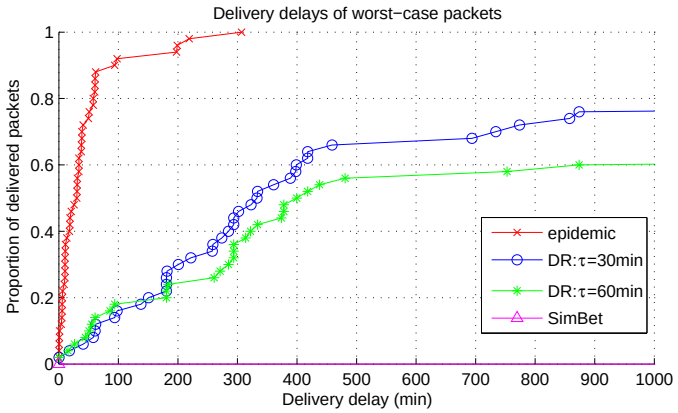
parameter	value
Number of nodes	30 – 50
communication range	80 m
packet size	1kB
number of testcases	100

3.4.2 Experiment 1: Duke Campus Trace

For the Duke trace, the simulation is run on 30 out of 54 users. For each test case, the source and destination nodes are two students selected from our surveyed trace who never encounter one another. We then randomly select another 28 intermediate students. The source starts periodically sending packets right after the system is warmed up. The packets are sent with an interval of 20 minutes, and the entire process lasts for 7 hours. We present our experimental results to evaluate the performance of DR in terms of the delivery delay and communication overhead in both (a) **the overall-case scenario** in which the destination node either stays in its usual cluster or in other clusters, and (b) **the worst-case scenario** in which the



(a) Case 1: The overall-case packets



(b) Case 2: The worst-case packets

FIGURE 3.8: Delivery ratios of packets vs delivery delay.

destination node deviates from its usual cluster.

Tunable Delivery Delay

In Figure 3.8, we show the delivery ratio over the delivery delay for all three protocols in the overall scenario and the worst-case scenario. We observe that, in both scenarios, the delivery delay of DR can be easily tuned between the delay of Simbet and the epidemic delay (optimal) by adjusting the value of τ.

Moderate Communication Overhead

Next, we compare the communication overhead between epidemic routing, DR, and SimBet. We use the average number of transmits per packet as the metric for evaluating the communication overhead. Again, we examine the average communication overhead for both scenarios. Figure 3.9 shows the comparisons under the same scenarios as in Figure 3.8. The communication overhead of DR can be moderate with a proper value of τ. For example, with $\tau = 60$ minutes, the communication overhead of DR is approximately 5 transmits/packet. Although this overhead is slightly larger than that of SimBet (approximately 2 transmits/packet), it is still acceptable. The DR protocol overhead falls between the SimBet and epidemic routing overhead, as expected. Also, we observed that whether the destination node is in its usual cluster or not has little impact on the communication overhead for both DR and SimBet. This observation is expected, because for both the overall case and worst case, DR always distributes packets to all of the possible social clusters, while SimBet always waits for the nodes to return to their usual clusters. Therefore, the two cases do not affect the communication overhead.

Worst-Case Behavior

We next revisit the worst-case behavior of DR. When comparing the delivery ratios of the packets in the worst-case scenario to those from the overall-case scenario, we see that the delivery ratio of epidemic routing and DR routing have not changed significantly; however, the delivery ratio of SimBet drops dramatically. For example, Figure 3.10(b) shows that in the worst cases, SimBet cannot deliver any of the packets within 1,000 minutes; however, for DR with $\tau = 60$ minutes, more than half of the packets can be delivered within 500 minutes. In Figure 3.9, we see that the communication overhead of DR and SimBet do not change significantly in the

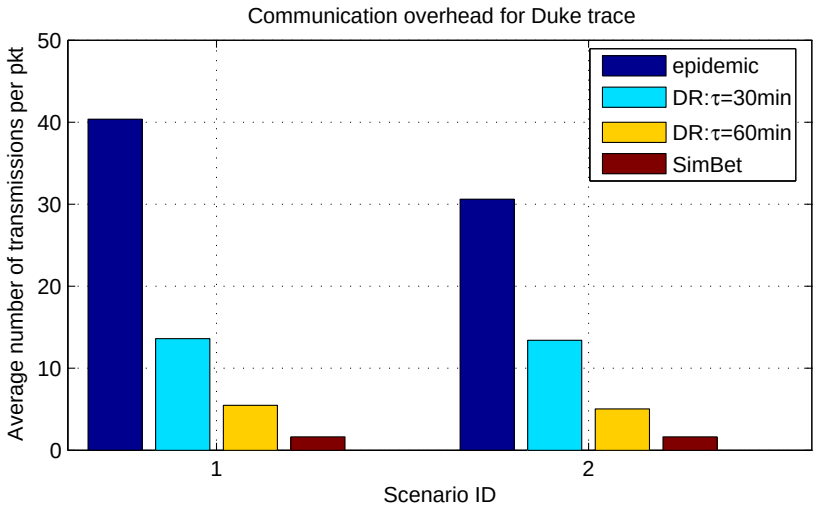


FIGURE 3.9: Communication overhead for the Duke trace. Scenario #1 is the overall case, and scenario #2 is the worst case.

worst-case scenario. Our experimental results show that, compared to SimBet, DR can tolerate the deviation of nodes from their usual activities.

Figure 3.9 also shows the overhead relative to τ for DR. We see that we can tune the size of the dominating set by tuning the value of τ, thus obtaining the desired communication overhead between that of SimBet and epidemic routing.

Trade-Off between Delay and Overhead:

We next study the delay-overhead tradeoff of the three algorithms. Figure 3.10 shows the average delivery delay versus the average communication overhead of DR for different values of τ. The delay-overhead tradeoff of SimBet and epidemic routing are included as well for comparison. Note that epidemic routing is actually a special case of DR – when $\tau = 0$, each node is categorized into a unique cluster and DR reduces to epidemic routing.

Our first observation on DR is that the delivery delay monotonically decreases as

τ decreases and saturates when $\tau < 30$ minutes. Additionally, the communication overhead monotonically decreases as τ increases and saturates when $\tau > 75$ minutes. From these results, we see that DR is flexible, as it can achieve any trade-off between epidemic routing and SimBet routing when the value of τ is chosen appropriately. Moreover, for the Duke traces, the impact of τ on the delivery delay and communication overhead is significant only when the value of τ is between 30 and 75 minutes.

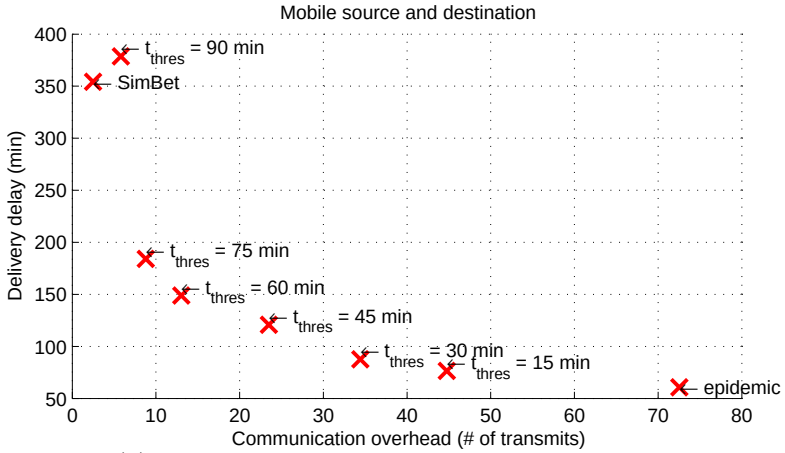
3.4.3 Experiment 2: WTD Trace

We run all of the protocols on a week's record of the WTD trace. Because the accurate locations of the nodes are not available in the trace, we use the location-free DR algorithm. We run our simulation on DTNs with 30, 40, and 50 nodes. For each size of DTN, we run 100 test cases. For each test case, we pick a random set of nodes and randomly choose two of them as the source and destination nodes. The source node begins sending packets to the destination with an interval of 20 minutes from 5×10^4th second and stops sending packets at 1.5×10^5th second. The encounters of the nodes here are more sparse than in the Duke trace. Therefore, we expect a longer delivery delay and we use larger values of τ.

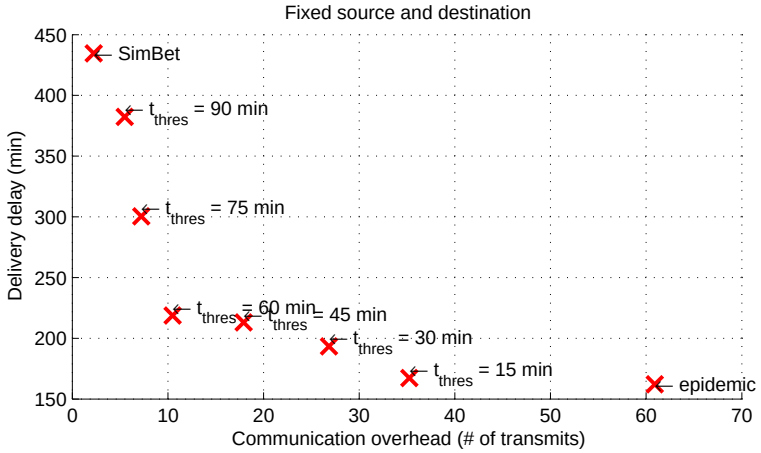
Tunable Delivery Delay

We first compare the packet delivery ratio between different numbers of nodes for the three protocols. Figure 3.11 shows the delivery ratio over the delivery delay for epidemic, DR, and SimBet, for DTNs with 30, 40, and 50 nodes.

Note that, for the same protocol, the gaps between the delivery delays of different sizes of DTNs are very small, especially for DR and SimBet. In other words, the



(a) Scenario 1: Mobile sources and destinations



(b) Scenario 2: Fixed source and destination

FIGURE 3.10: Delivery ratios of packets vs delivery delay.

density of nodes has only a minimal impact on DR and SimBet in terms of the delivery delay. We explain this observation as follows: DR and SimBet exploit the social clustering property of human mobility. While DR tries to distribute the packets to all of the possible clusters, SimBet attempts to forward the packet to the destination's cluster. Therefore, both algorithms are mainly affected by the

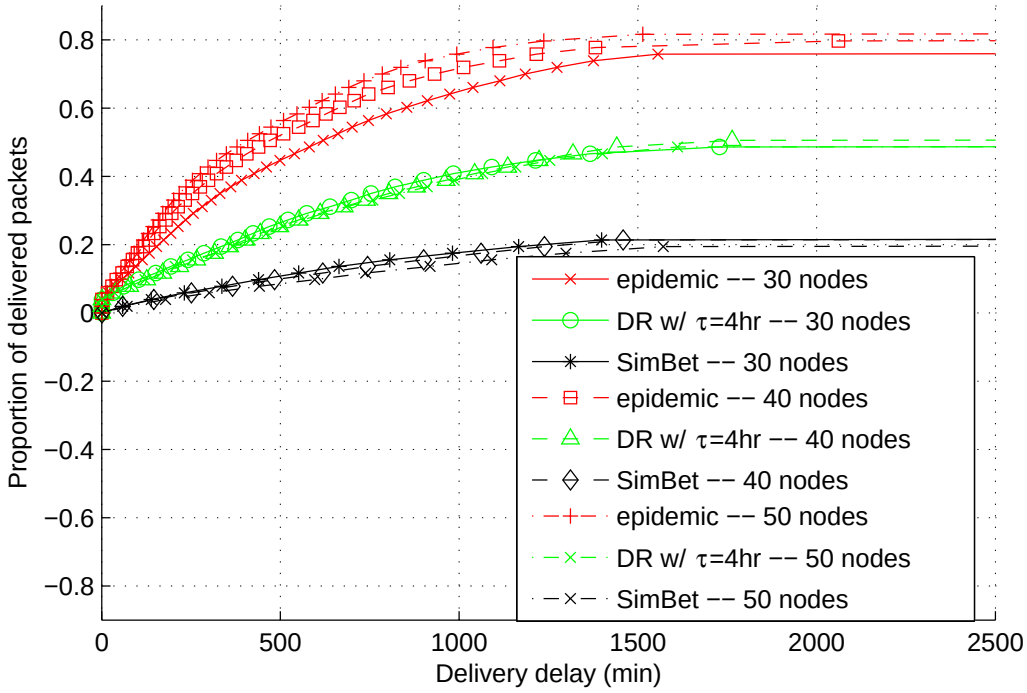


FIGURE 3.11: Delivery ratios of packets vs. delivery delay for all protocols and numbers of nodes (WTD trace).

delay between different clusters. On the other hand, when we increase the density of nodes, we are more likely to increase the number of nodes in each social cluster than to increase the number of social clusters. In this case, increasing the overall density of nodes should not significantly impact the delivery delay. On the other hand, epidemic routing attempts to distribute the packets to all of the nodes instead of exploiting any social clusters. Therefore, the impact of node density on epidemic routing is expected to be larger than DR and SimBet. When the number of nodes increases from 30 to 50, we observe that the delivery delays of DR and SimBet only slightly change, as shown in Figure 3.11. This is because both routing algorithms are

exploiting the social clustering property of human mobility and are mainly affected by the delay between different clusters. When we modify the number of nodes, the size of each social cluster is more likely to change than the number of social clusters. Therefore, changing the overall density of nodes should have only a small impact on the delivery delay. In this case, we only show the results for 40 nodes in Figure 3.12, and the other results are omitted. In Figure 3.11, we observe a large proportion of undelivered packets even if we use the epidemic routing until the end of the simulation – this can be attributed to the long time interval between node encounters. In Figure 3.12, we ignore the undelivered packets.

From these results, we again observe that DR outperforms SimBet in terms of delivery delay for the WTD trace. In Figure 3.12(b), we show the packet delivery ratio for the worst-case scenario. We note that DR can handle these packets much better than SimBet and that the difference between DR and SimBet is greater than in the Duke trace.

Communication Overhead:

We next compare the communication overhead of the protocols for UCSD WTD trace and show the results of comparison between different protocols and different numbers of nodes in Figure 3.13. As we did in the Duke trace, the average number of transmits for each packet is used as the metric for the communication overhead.

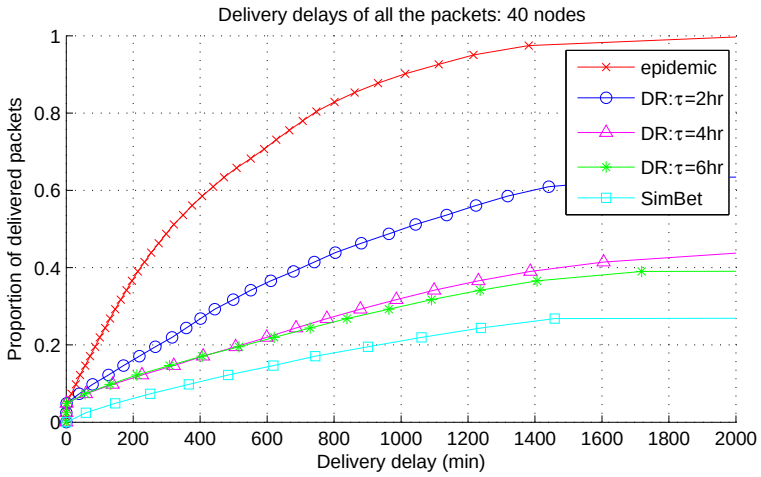
Our observations from the WTD trace matches our conclusion drawn from the Duke trace – DR is flexible because 1. its delivery delay can be tuned by adjusting τ, 2. its communication overhead is only slightly larger than that of SimBet, and 3. it is resilient to unusual movements of nodes. Similar to the Duke trace, the communication overhead of DR and SimBet do not change significantly, irrespective of whether

the destination deviates from its usual cluster. However, the communication overhead of DR is closer to SimBet compared to the Duke trace. We also observe that the communication overhead of epidemic routing gradually increases as the number of nodes increases, which is expected. On the other hand, the communication overhead of DR almost remains the same when the number of nodes increases.

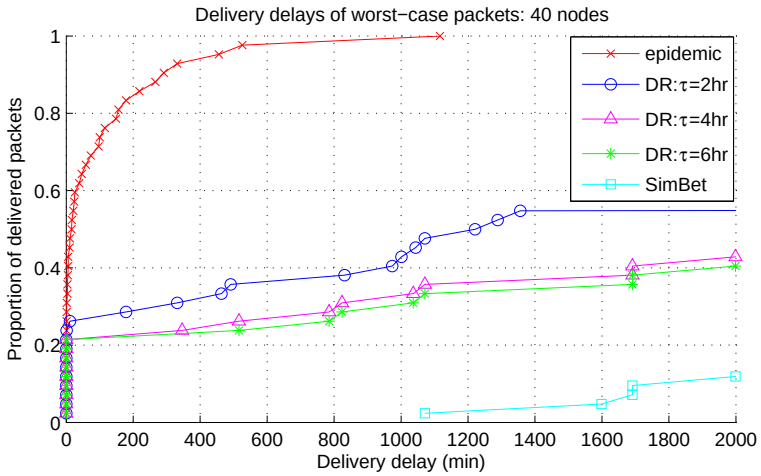
3.5 Conclusions

In this chapter, we have proposed the DR protocol, a novel routing protocol for DTN that exploits the cluster properties of mobile networks composed of people. In this protocol, each packet is multicast to all of the clusters to achieve a limited delay. Using this approach, the delivery delay is made comparable to the optimal delay that is achieved by epidemic routing and the communication overhead is much lower than that for epidemic routing. We have also shown that the delivery delay can be tuned by choosing different parameter values. We demonstrated the performance of DR on real-life traces, and the experimental results show that the performance of DR is almost unaffected by the irregular behavior of nodes. Additionally, the communication overhead of DR is moderate. We have shown that this approach provides a flexible and stable method for message propagation in a DTN.

Diverse routing can be made even more scalable if we can incorporate GLS routing (Li et al., 2000) to make diverse routing hierarchical, which is left for future work. Another important future direction is to add a congestion-control mechanism onto the routing protocol such that the packets are scattered to different nodes rather than only to the dominating set, thus preventing the dominating set from being a bottleneck in data dissemination.

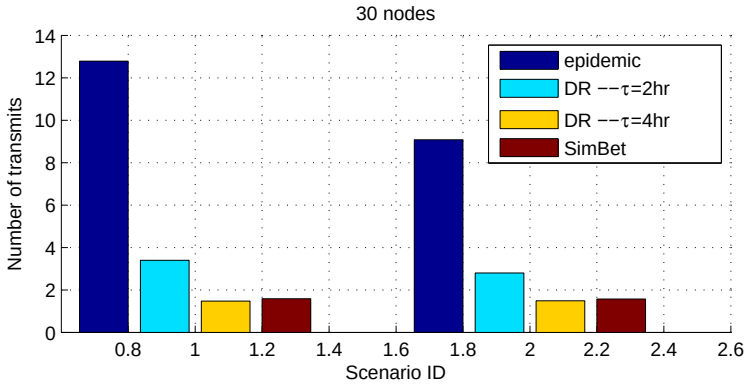


(a) Case 1: The overall-case packets

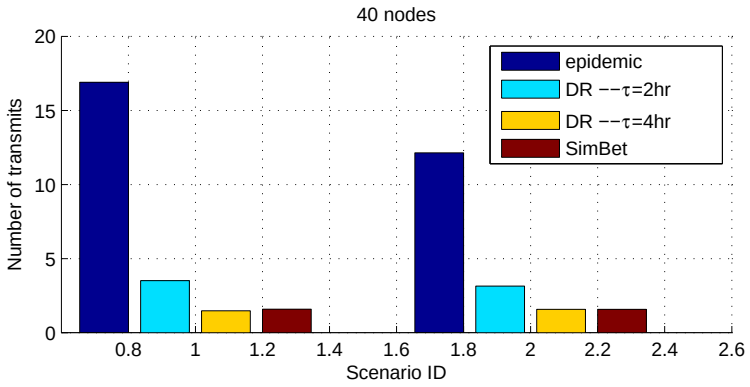


(b) Case 2: The worst-case packets

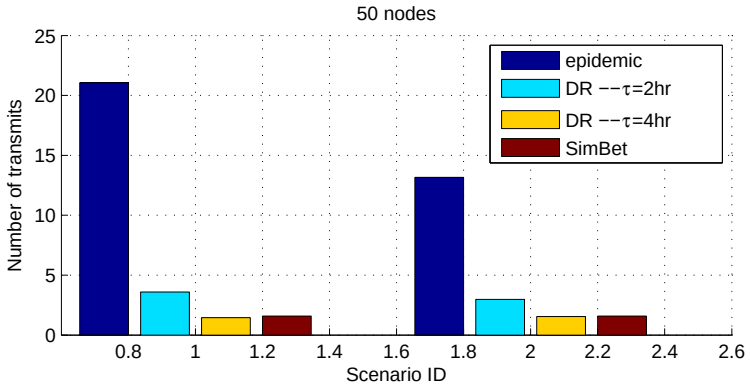
FIGURE 3.12: Delivery ratios of packets vs. delivery delay (WTD trace).



(a) DTN with 30 nodes



(b) DTN with 40 nodes



(c) DTN with 50 nodes

FIGURE 3.13: Communication overhead under the WTD trace. Scenario #1 refers to the overall case, and Scenario #2 refers to the worst case.

4

P^{2}DAP: Privacy-Preserving Detection of Sybil Attacks

In this chapter, we propose a privacy-preserving scheme to detect Sybil attacks in VANETs with a commonly used framework from existing work (Raya and Hubaux, 2005). The framework assumes that vehicles communicate with one another in a multihop manner and that the communication is monitored by an RSU through passive overhearing.

The rest of the chapter is organized as follows. Section 4.1 describes the system model and assumptions. Section 4.2 presents the proposed detection scheme for handling specific requirements. Section 4.3 further discusses the P^{2}DAP algorithm and proposes three possible improvements. Section 4.4 presents simulation results. Finally, Section 4.5 concludes the chapter and outlines future research directions.

4.1 System Model

In this section, we describe our assumptions regarding the VANET system, the capabilities of the attackers, and the Sybil attacks.

4.1.1 Assumptions on VANET Architecture

1. The *DMV* is the trusted party that maintains vehicle records and distributes certified pseudonyms to vehicles. The DMV has enough resources to generate pseudonyms quickly and store all of the vehicle-related information, and it is consulted when any authoritative clarification is necessary. However, these DMV services are not designed for heavy network traffic – excessive communication can turn the DMV into a bottleneck.

2. *Vehicles* are untrusted parties. They communicate with each other in a multihop manner. A message exchanged among vehicles is signed with a DMV-certified pseudonym.

3. *RSUs* are wireless access points. They are scattered along the road and connected to the DMV via a backhaul network, acting as intermediaries to the DMV. The RSUs monitor vehicular activity, identify suspicious behavior, and report to the DMV for confirmation and punishment. The RSUs may be compromised and thus cannot be used for critical functions – for example, the RSU cannot authenticate a message or distribute pseudonyms. However, they can be used to improve the scalability of a system.

4.1.2 Assumptions of Attackers' Actions

In this section, we discuss the actions of the attackers of interest.

1. *Announce false messages – False data injection:* A vehicle can sign a false message and then broadcast it. Such an attack cannot be identified from the message itself, as the message can be signed with a CA-certified pseudonym. This attack may be handled by a majority voting scheme if there are more benign vehicles than attackers. However, the voting scheme fails if the attacker carries out a Sybil attack by generating sufficient false identities to outvote benign vehicles, which we will discuss next.

2. *Pretend to be multiple vehicles by using multiple pseudonyms – Sybil attack:* In a VANET, a vehicle can carry out a Sybil attack by using its multiple pseudonyms to sign messages. If a vehicle can be identified from a set of pseudonyms it uses, then the vehicle's privacy is compromised. As a result, vehicles and RSUs that overhear multiple messages signed with an attacker's pseudonyms have no means of recognizing that these pseudonyms actually belong to the same vehicle. This thesis proposes a method to solve this problem.

3. *Compromised RSUs:* RSUs are semi-trusted parties, and some RSUs may be compromised by the attackers. We assume that a compromised RSU can be easily detected by the DMV, thus being quickly revoked. However, attackers can still gain information stored in the RSUs. Therefore, a scheme's resilience to RSU compromise is determined by the amount of information released to and held by the RSUs.

In summary, it is necessary to have a framework that allows RSUs to detect a Sybil attack without knowing the association between pseudonyms and unique vehicle IDs (i.e., without compromising privacy). This framework needs to be scalable in terms of the workload it imposes on the DMV, and it needs to be robust against RSU compromise. Moreover, it is important to quickly detect the attack and perform subsequent punishments/revocations to minimize the impact of the attack. To address these concerns, we propose a new scheme, referred to as Privacy-Preserving Detection of Abuses of Pseudonyms (P^2DAP).

4.1.3 Structure of Events and the Use of Pseudonyms

In vehicular network applications, vehicles are expected to broadcast specific events whenever they observe them. Counting the number of vehicles that send the same message is an important primitive upon which several VANET applications depend. To clarify the notion of *same* or *different* events, we must unambiguously define the format of "events".

An event is a tuple (t, l, e) *generated at a pre-defined time interval* $t \in \mathbf{T}$ *in a pre-defined region* $l \in \mathbf{L}$ *for an event type* $e \in \mathbf{E}$*, where* $\mathbf{T}, \mathbf{L}, \mathbf{E}$ *are distributed to vehicles.*

An attacker carries out a Sybil attack by abusing multiple pseudonyms. On the other hand, in order to avoid being tracked, benign vehicles also use multiple pseudonyms to report events. To distinguish the benign use of pseudonyms from the abuse of pseudonyms, we now introduce the following restriction on the use of pseudonyms.

A benign vehicle can use only one pseudonym to sign one event.

If a vehicle uses multiple pseudonyms to sign an event such that others believe there are multiple vehicles reporting the same event, the action is considered to be a *Sybil attack*, and the vehicle is deemed to be malicious.

4.2 Proposed P^{2}DAP Scheme

This section presents our technique for handling Sybil attacks. The main purpose is to detect Sybil attacks and revoke malicious vehicles immediately after detection. A baseline method to detect a Sybil attack is to forward all of the reported events to the DMV and have the DMV examine the signatures of each message. On observing a single event (t_i, l_j, e_k) signed with two different pseudonyms from the same vehicle, the DMV considers that vehicle to be an attacker. The drawback of this method is the heavy network traffic on the DMV. Therefore, we propose the P^{2}DAP scheme, which uses RSUs to perform most of the DMV's tasks and thereby reduces the communication overhead. We also discuss how our scheme preserves privacy in the case of RSU compromise.

4.2.1 Complete Two-Stage P^{2}DAP Scheme

We first propose the Complete Two-Stage P^{2}DAP Scheme, abbreviated as C-P^{2}DAP. Later, we will propose a number of variants of C-P^{2}DAP that improve the performance of the scheme. In the P^{2}DAP scheme, we delegate most of the detection to RSUs and involve the DMV only when suspected vehicles must be confirmed as Sybil attackers. However, because RSUs are not trusted entities, the vehicle information available to the DMV cannot be transferred to the RSUs. In view of these constraints, we divide the vehicles into groups and release the group information to

RSUs. This information allows RSUs to detect suspicious behavior, but is not sufficient for RSUs to just track the vehicles because RSUs cannot distinguish a vehicle from a group of vehicles. To group the vehicles, we use the one-way hash function to hash the pseudonyms during initialization.

Initialization Step

Initially, the DMV knows the total number of vehicles and sequentially generates a sufficient number of yearly pseudonyms for all of the vehicles. After generating a pseudonym p, the DMV first hashes $(p \mid \kappa_c)$ using a one-way hash function, where κ_c is a global key. The DMV then selects a set of bits from the hashed result to create the hash collisions. The selected bits are referred to as the "coarse-grained hash value". After that, the pseudonym p is placed into a group, which stores the pseudonyms with the same coarse-grained hash values. In other words, for each pseudonym p_l in the m-th coarse-grained group, we have $H(p_l|\kappa_c) = \Gamma_m$, where H is a one-way hash function and Γ_m is the coarse-grained hash value for group m. We refer to such groups as "coarse-grained groups". The key κ_c is distributed to all the RSUs.

Next, the DMV calculates the hash value for the above p with a new key κ_f and selects a set of bits from the result. The bits selected from the new hash value are referred to as the "fine-grained hash value". The pseudonym p is then placed into a subgroup of the coarse-grained group, called the fine-grained group, in which all of the pseudonyms have the same fine-grained hash value. For each pseudonym $p_{l'}$ in the n-th fine-grained group in the m-th coarse-grained group, we have $H(p_{l'}|\kappa_f) = \Theta_n$, where Θ_n is the fine-grained hash value for the subgroup n.

The above steps are referred to as a "two-level hash" and are shown in Figure

4.1. The DMV continues to generate and to two-level hash pseudonyms until all of the fine-grained groups contain enough pseudonyms for a vehicle's use. Then, the DMV loads a unique fine-grained group of pseudonyms on each vehicle at the time of the yearly vehicle registration and stores the corresponding $(\Gamma_m|\Theta_n)$ as the vehicle's secure plate number. From the above description, it is obvious that the mapping from secure plate numbers to vehicles is one-to-one. Thus, the DMV must carefully choose the length of Γ_m and Θ_n such that the total number of available secure plate numbers are greater than or equal to the number of vehicles.

The fine-grained hashing step provides a measurement of privacy to the vehicles when an RSU is compromised. Details about the level of privacy are discussed in Section 4.4.3. The two-level hashing saves storage for the DMV, because the DMV can link a pseudonym to a vehicle by calculating its coarse-grained and fine-grained hash values and then comparing them with the secure plate number $(\Gamma_m|\Theta_n)$. This obviates the need to maintain a table of vehicle secure plate numbers and the corresponding pseudonyms.

After the initialization stage, the DMV stores the secure plate number for each vehicle and secretly keeps the fine-grained hash key κ_f.

Generating Pseudonyms with Short-Period Keys

When generating the pseudonyms, we must consider the lifetime of a coarse-grained key κ_c because an attacker gaining access to an RSU can partially learn the pseudonyms of all of the vehicles for that lifetime. If the lifetime is too long, the privacy of the vehicles will be severely impaired. Therefore, κ_c should be given a short lifetime, such as one or two days. We can then modify the initialization stage as follows.

We divide an entire year's time into Ω time intervals and take the time intervals to

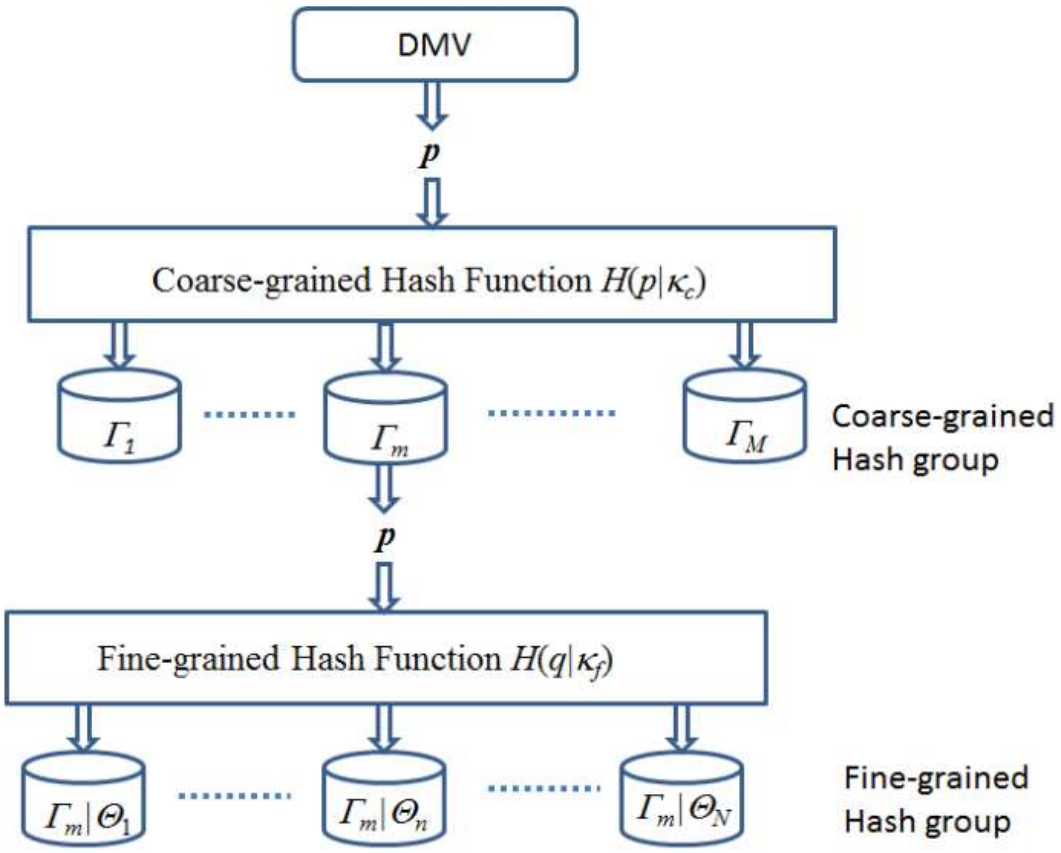


FIGURE 4.1: The generation and two-level hashing of a pseudonym p.

be one day. In the initial pseudonym generation, the DMV uses a set of coarse-grained keys K_c instead of one key, κ_c, to hash the pseudonyms. Each key $\kappa_{c,\gamma} \in K_c$ is used to generate pseudonyms for the γ-th time interval. The pseudonyms that are hashed to Γ_m with the key $\kappa_{c,\gamma}$ will be put into the m-th coarse-grained group, and they can only be used in the γ-th time interval. After that time interval, these pseudonyms will be discarded. To prevent malicious vehicles from using expired pseudonyms to sign events, the DMV uses different keys to generate certificates for pseudonyms

for different time intervals. Thus, vehicles can recognize an expired pseudonym by examining its certificate. Initially, the DMV secretly holds all of the coarse-grained hash keys. At the beginning of the γ-th time interval, the DMV releases the key $\kappa_{c,\gamma}$ to the RSUs. With this approach, an RSU holds each valid coarse-grained key only for a short time. When an RSU is compromised, the attacker only obtains the coarse-grained hash key for the current time interval. We do not impose any restrictions on the fine-grained key κ_f, because the DMV does not release it and an attacker cannot obtain it by compromising an RSU. Key κ_f can either be the same key over the registration period or it can have different values for each day, similar to κ_c.

Compared to the long-period keys, this short-period key generation uses Ω coarse-grained hash keys instead of one, thus bringing extra storage overhead to the DMV. At the same time, the computational overhead will increase, which we will show explicitly in Section 4.4.2. The DMV secretly holds all of the coarse-grained hash keys initially and gradually releases the key $\kappa_{c,\gamma}$ to the RSUs.

Sybil Attack Detection Step

When vehicles communicate, an RSU overhears all of the vehicles within its communication range and puts the pseudonyms used to sign the event (t_i, l_j, e_k) in the list $L_{i,j,k}$. When all of the pseudonyms for the event (t_i, l_j, e_k) are collected, the RSU detects the Sybil attacks as follows. The RSU iterates through each pseudonym p in the list $L_{i,j,k}$ and computes the coarse-grained hash value $H(p|\kappa_c)$. (Recall that κ_c is pre-distributed to all RSUs in the initialization step.) If $\exists p, p' \in L_{i,j,k}$ such that $H(p|\kappa_c) = H(p'|\kappa_c)$, then the RSU notices that two pseudonyms of the same coarse-grained hash value are used to sign the event (t_i, l_j, e_k). This can be either (i) a Sybil

attack where one vehicle is using multiple pseudonyms to report the same event, or (ii) a *false alarm*, where an event is reported by two vehicles whose pseudonyms are in the same coarse-grained group. The RSU cannot discriminate between (i) and (ii) and it sends the report to the DMV. The report contains the event (t_i, l_j, e_k), the hash value Γ, the pseudonyms whose coarse-grained hash value is Γ, the signatures of the event, and the certificates accompanying the pseudonyms.

On receiving an RSU report, the DMV first verifies the signatures and the coarse-grained hash value Γ to prevent a compromised RSU from implicating a benign vehicle. If the RSU proves to be honest, the DMV calculates the fine-grained hash value $H(p|k_f)$ for each pseudonym p in the RSU report. If $\exists p, p'$ in the report such that $H(p|\kappa_f) = H(p'|\kappa_f)$, the DMV concludes that p and p' are from the same vehicle and that it has attempted a Sybil attack. The DMV then takes further action to punish or revoke the malicious vehicle.

In this scheme, a Sybil attack is guaranteed to be detected. However, when the vehicles are densely distributed, false alarms can happen often. The issue cannot be solved by simply increasing the number of coarse-grained hash values because that can violate the privacy of vehicles.

For example, the format of pseudonym p can be $\{\text{date}|\text{random number}_j\}$, and the format of the day can be month|day. For each day_i, the DMV computes the hash values of all of the pseudonyms $\{\text{day}_i|\text{random number}\}$, represented as $\{p_{ij}\}_{j=1,2,\cdots}$, that are concatenated with k_{ci}; that is, $H(p_{ij}|k_{ci})$.

When the DMV distributes a pseudonym set P_r to vehicle r, we have

$$\forall p_{ij_1}, p_{ij_2} \in P_r, H(p_{ij_1}|k_{ci}) = H(p_{ij_2}|k_{ci}) \tag{4.1}$$

for the pseudonyms that will be used in a given day i.

However, the hash values for the different days may be different.

The use of short-lifetime keys, instead of one key, provides several benefits. First, if an RSU is compromised, the attacker gains less information because the current key will expire after one day instead of one year. Second, it also provides a feasible and straightforward method for immediate pseudonym revocation. In order to immediately revoke a malicious vehicle, the DMV only needs to broadcast all of the pseudonyms of the malicious vehicle that are valid for that day. Because the number of pseudonyms for one day is not large, it is feasible to broadcast the pseudonyms of a revoked vehicle to other vehicles. Benign vehicles can then discard messages authenticated by these pseudonyms.

4.2.2 E-P^{2}DAP – Detecting Events Instead of a Sybil Attack

In the C-P^{2}DAP scheme, an RSU reports to the DMV whenever it finds any set of pseudonyms that hash to the same coarse-grained values. Thus, when an event is reported by a large number of vehicles, C-P^{2}DAP can cause false alarms. Clearly, on a road with heavy traffic, such false alarms will create a heavy communication overhead on the DMV. To address this problem, we observe that detecting each and every Sybil attack may not be necessary for practical VANET applications. If a Sybil attacker does not report an event conflict to other benign vehicles, it is not always necessary to detect the attacker. In other words, there are cases in which an attacker can only cause harm by broadcasting false data, thus misleading other vehicles and RSUs. We first make the following assumptions: (i) each false (faked) event is generated by only one malicious vehicle; (ii) benign vehicles will not report

false events. We then propose the Event-P^{2}DAP scheme (abbreviated as E-P^{2}DAP), which does not detect all Sybil attacks but only detects those creating false events.

For an event (t_i, l_j, e_k), the RSU collects a list of pseudonyms $L_{i,j,k}$ used to sign the event. If $\forall p, p' \in L_{i,j,k}$, $H(p|\kappa_c) = H(p'|\kappa_c)$ (i.e., all the pseudonyms used to sign (t_i, l_j, e_k) have the same coarse-grained hash value), then the event is probably sent from only one vehicle and is likely a faked event. However, it can also be a coincidence that all of the reporting vehicles happen to have pseudonyms with the same coarse-grained hash value. In this case, the RSU generates a report with the same format as in C-P^{2}DAP and sends it to the DMV. The DMV performs a further check on these pseudonyms to see if they belong to the same vehicle by computing their fine-grained hash values.

In this scheme, the DMV only needs to examine the pseudonyms in two cases: 1) if an attacker reports a false event and carries out a Sybil attack; 2) if a true event is reported by multiple benign vehicles whose pseudonyms have the same coarse-grained hash value, which is a false alarm. Obviously, the number of false alarms is likely to be small compared to the total number of the pseudonyms that RSUs process. Therefore, the RSUs are able to efficiently appropriate most of the pseudonym processing tasks, thus reducing the burden on the DMV. The proof is shown as follows. Assume that, for an event e_i, the number of vehicles reporting e_i is N_V, the number of hash values of pseudonyms is N_H, the hash values are uniformly distributed among vehicles, and all of the vehicles are benign. Denote the probability of false alarms at any time as P_{FA}. It can be easily seen that $P_{FA} = (1/N_H)^{N_V - 1}$, therefore $\lim_{N_V \to \infty} P_{FA} = \lim_{N_V \to \infty} (1/N_H)^{N_V - 1} = 0$. Simulation results in Section V confirm these conclusions for a wide range of scenarios.

4.2.3 T-P²DAP – Detecting Collusion

One issue with E-P²DAP is that it cannot detect colluding vehicles, i.e., two or more malicious vehicles reporting the same faked event. To address collusion, we propose Threshold-P²DAP (abbreviated as T-P²DAP), described as follows: we assume that each faked event is generated by a small number of colluding attackers instead of one attacker but that that number will not exceed a threshold τ. Then, for a pseudonym list $L_{i,j,m}$, the RSU calculates the coarse-grained hash value for each pseudonym $p \in L_{i,j,k}$ and obtains a set of coarse-grained hash values S_c. If $|S_c| \leqslant \tau$ and two or more pseudonyms in $L_{i,j,k}$ map to the same coarse-grained hash value, the RSU suspects the event to be fake and reports to the DMV. Similar to C-P²DAP and E-P²DAP, the DMV in T-P²DAP then examines the RSU report and determines whether the event is from attackers. If two or more of the pseudonyms have the same fine-grained hash value, the DMV concludes that there is a Sybil attack and it revokes the vehicle.

Compared to E-P²DAP, T-P²DAP is more resilient to collusion. Any false event reported by less than τ attackers can be detected by an RSU. Obviously, T-P²DAP has a larger false alarm rate than E-P²DAP. Therefore, choosing a proper value of τ to achieve a tradeoff between detection accuracy and communication overhead is important.

In this section, we introduced C-P²DAP, E-P²DAP, and T-P²DAP. Note that both C-P²DAP and E-P²DAP are special cases of T-P²DAP. In T-P²DAP, if $\tau = 1$, an RSU detects a Sybil attack by verifying that the coarse-grained hash values of the pseudonyms used to sign a single event are the same. Thus, T-P²DAP reduces

to E-P^{2}DAP. On the other hand, when τ is greater than or equal to the number of coarse-grained hash values, an RSU reports a Sybil attack once it finds the coarse-grained hash values of two pseudonyms used to sign an event are the same. In this case, T-P^{2}DAP reduces to C-P^{2}DAP. Currently, T-P^{2}DAP cannot detect colluding vehicles if each malicious vehicle only reports a faked event with one pseudonym. However, such an attack is no longer a Sybil attack and is beyond the scope of our discussion. Section 4.4 contains more details of the performance comparison between the above three variants of P^{2}DAP.

4.3 Discussions: Improvements on P^{2}DAP

In this section, we propose several improvements to the above P^{2}DAP scheme from the perspectives of key revoking convenience and adaptivity.

4.3.1 Revoking the Pseudonyms of Malicious Vehicles

After a malicious vehicle is detected, the DMV should revoke all of its pseudonyms. In this subsection, we discuss three possible approaches for revoking vehicles that can be combined with P^{2}DAP.

The first two approaches are to use the revocation schemes proposed in (Raya et al., 2006), i.e., Revocation of the Tamper-Proof Device (RTPD) and Revocation using Compressed Certificate Revocation Lists (RC^{2}RL). The RTPD requires hardware support on the vehicle; specifically, a tamper-proof device (TPD) is installed to store pseudonyms and sign messages. On observing a malicious vehicle, the DMV sends a revocation message to the TPD, at which point the TPD will erase all of the pseudonyms and stop signing messages. With this approach, the DMV can revoke a

vehicle with a single message.

Contrary to RTPD, RC^{2}RL does not assume any hardware support such as the TPD; instead, it creates a bloom filter for all of the pseudonyms to be revoked. The bloom filter is then broadcasted to all the vehicles. On receiving a message, a vehicle uses the bloom filter to verify its pseudonym and drop it if the pseudonym is determined to have been revoked. In the paper (Raya et al., 2006), the size of each bloom filter is estimated as tens of Kbytes. Therefore, in order to revoke a vehicle, the DMV must flood tens of Kbytes throughout the network.

The third approach is to create a secret key for each vehicle that helps to identify its pseudonyms. If an entity knows the secret key of a vehicle, it is able to recognize all of the pseudonyms of that vehicle. Such a secret key is referred to as a "backdoor". The Group Signature (GS) scheme proposed in (Calandriello et al., 2007) can be used to generate such a backdoor. In the GS scheme, each vehicle is equipped with a group public key gpk_{CA} and a private signature key gsk_V and generates its own pseudonyms. On creating a pseudonym K_V^i, the vehicle generates a group signature $\Sigma_{CA,V}(K_V^i)$ for K_V^i with the private key gsk_V and then calculates a certificate $Cert_{CA}^H(K_V^i)$ using the group public key gpk_{CA}. Other vehicles can verify K_V^i by validating $\Sigma_{CA,V}(K_V^i)$ using $Cert_{CA}^H(K_V^i)$. When the DMV needs to revoke a vehicle, it simply broadcasts the vehicle's private signature key gsk_V.

The idea of GS is adopted in the initialization stage of P^{2}DAP, where the DMV generates gsk_V and $\Sigma_{CA,V}(K_V^i)$ for the vehicle V when generating a pseudonym K_V^i. The DMV will distribute the pseudonyms, the group signatures, and the group public key gpk_{CA} to the vehicles and keep the private key gsk_V as the "backdoor" of V. The vehicles can then use the pseudonyms to sign the messages and use gpk_{CA}

to verify the certificates of the pseudonyms. When the DMV needs to revoke a vehicle V, it broadcasts gsk_V, the same as in the GS signature scheme. On receiving gsk_V, a vehicle can verify whether a pseudonym is from any other vehicle V and is thus able to identify the messages from a revoked vehicle. The GS algorithm used in (Calandriello et al., 2007) requires 8 modular exponential operations for each signature. With such a number, we estimate the computational cost of generating pseudonyms with GS to be approximately three times the cost of using RSA public key cryptography to generate pseudonyms, as is done in Section 4.2.1.

In this subsection, three different approaches for revoking pseudonyms are discussed. In these approaches, RTPD incurs small communication and computation overhead but requires hardware support. RC^{2}RL and GS do not need hardware support, but they respectively require large communication overhead and computation overhead. Each method has its pros and cons, and the "best" method to choose will depend on a system's available resources.

4.3.2 τ-P^{2}DAP: Real-Time Adaptive P^{2}DAP Scheme

This subsection discusses a P^{2}DAP scheme that is adaptive to real-time traffic. For a particular RSU, the number of nearby vehicles is a time-varying value. According to the study in (Suh et al., 2005), within one day, the traffic volume in a street can range from 10 vehicles/hour (at midnight) to 3000 vehicles/hour (at peak hours). These fluctuations in the traffic volume cause difficulty in using a single variant of P^{2}DAP to efficiently detect the attackers. When there are fewer vehicles, the E-P^{2}DAP or the T-P^{2}DAP with a small threshold is preferred so that malicious vehicles can be detected with a smaller cost. When the traffic volume is high, either the C-P^{2}DAP or

the T-P^{2}DAP with a large threshold is selected to better catch collusions. A method to solve this problem is to make each RSU adaptively choose the detection scheme based on the traffic volume.

To this end, we propose τ-P^{2}DAP, in which each RSU checks the total number of received packets for each reported event (written as N_{PE}) when it attempts to detect a Sybil attack. N_{PE} can then be a parameter for calculating the value of τ in T-P^{2}DAP, using the equation $\tau = \alpha N_{PE}$, where α is the estimated proportion of attackers among all of the vehicles. The value of α can be either pre-distributed to the RSUs or learned by the RSUs through the detection of attackers. Using α, we estimate the number of nearby attackers and use the estimation as the value of τ.

With τ-P^{2}DAP, the RSU is expected to adaptively report to the DMV based on the current traffic status. The algorithm may not always be the optimal algorithm if the malicious vehicles intentionally mislead the RSU. For example, the malicious vehicles can use an extremely large number of pseudonyms to sign an event, which would force the RSU to use C-P^{2}DAP and incur huge communication overhead for the DMV. Additionally, several colluding malicious vehicles can generate many events, each signed with a small number of pseudonyms, causing the RSU to use E-P^{2}DAP and miss the events. However, the malicious vehicles are guaranteed to be quickly detected in the first case, and the discussed large communication overhead will hence only last for a short period. In the second case, these faked events can be easily filtered out when there is a large enough number of benign vehicles around.

4.3.3 κ-P^{2}DAP: Distribute Different Information to Different RSUs

This subsection discusses the adaptive P^{2}DAP algorithm from the location perspective. Even in the same city or county, the traffic volume varies significantly on different roads. As shown in a survey of traffic volume in 2002 at Columbia, NY (of Transportation, 2002), on different streets in the Columbia county, the average traffic volume ranges from 100 to 25,000 vehicles/day. Thus, it is difficult to determine the number of coarse-grained groups when P^{2}DAP is applied. A small number of coarse-grained groups will result in many false alarms on a highway, while a large number can harm the privacy of vehicles in a small community. To solve this issue, we propose κ-P^{2}DAP, which extends the 2-level hash keys to n-level hash keys to cope with different traffic volumes. In this scheme, the DMV distributes different subsets of the hash keys to the RSUs based on the nearby traffic volume.

When generating a pseudonym, instead of using two hash keys, the DMV uses q hash keys $\{\kappa_1, \cdots, \kappa_q\}$ to calculate the hash values. With each hash key κ_i, the DMV calculates an ϵ-bit hash value $\Gamma_{i,j} = H(p_j|\kappa_i)$ for the pseudonym p_j. The hash value for pseudonym p_j is written as $\mathbf{\Gamma_j} = (\Gamma_{1,j}, \cdots, \Gamma_{q,j})$ and can be considered as an element of a q-dimensional space $\mathcal{V}$ in which each dimension has 2^ϵ elements. The DMV continues to generate pseudonyms until their hash values fill the space $\mathcal{V}$.

Once all of the pseudonyms are generated, the DMV distributes the pseudonyms that can hash to the same value with the keys $\{\kappa_1, \cdots, \kappa_\beta\}$ to the same vehicle, where β is chosen such that $2^{\beta\epsilon} \geqslant N_V$. Then, the DMV adaptively releases the last δ hash keys $\{\kappa_{q-\delta+1}, \cdots, \kappa_q\}$ to the RSU, where δ is determined by the traffic volume around the RSU. If we want the RSU to be unable to distinguish vehicles passing

by within an hour, we should set $\delta = \frac{\log_2 K_V}{\epsilon}$, where K_V is the number of vehicles passing by the RSU in an hour. This indistinguishability of one vehicle among N multiple vehicles is defined as $N - anonymity$ (Sweeney, 2002) and is an important metric of privacy. More details about anonymity will be discussed in Section 4.4.3.

One possible issue of κ-P^{2}DAP is that an attacker can compromise an RSU that watches heavy traffic and use the knowledge to track vehicles on roads with less traffic. In this scenario, the vehicles will lose their privacy. Therefore, when releasing keys to an RSU, the DMV must consider both the local traffic and the cost of compromising the RSU. RSUs with more hash keys are expected to be more difficult to compromise. Fewer (1 or 2) hash keys are released to RSUs that are easier to compromise or are on roads with little traffic. κ-P^{2}DAP has increased computational overhead for the DMV and the RSUs. However, the communication overhead remains the same as the other P^{2}DAP variants discussed in Section 4.2.

4.4 Performance Evaluation

In this section, we evaluate the performance of P^{2}DAP using the following metrics: computation/communication overhead of the DMV, privacy of the vehicles, and detection latency. We also examine the tradeoff between detection latency and overhead. Results from the evaluation are expected to offer insights into the design of practical vehicular networks.

4.4.1 Simulation Setup

The P^{2}DAP scheme is simulated in ns-2 version 2.29. We used the 802.11a MAC and PHY layer protocol and the SHA-1 hashing as our hash function. In the initialization

Table 4.1: Parameters used in simulation.

Params	Values
Street length (m)	2,000
Comm Radius (m)	200
Street Width (lanes)	3
Lane Width (m)	3
Vehicle Speed (m/s)	25 – 35
Pseudonyms/Vehicle per Day	20
Vehicle Packet Rate (pkts/s)	3
Simulation Time (s)	400

stage, we used the SHA-1 hashing function to generate enough pseudonyms for all vehicles' use. The major simulation parameters are listed in Table 4.1.

In our simulation, we randomly generate events with different time intervals, locations, and types and then store them in a global array. When generating the events, we take the length of the time interval to be 20 seconds and the length of the location segment to be 250 meters and there is a total number of 5 different event types. Each vehicle periodically accesses the array, obtains the events with the current time interval and the vehicle's current location, and broadcasts the events. A vehicle also relays events heard from other vehicles.

We have defined four types of nodes: benign vehicles, malicious vehicles (attackers), RSUs and the DMV. A benign vehicle frequently senses the events, signs, and broadcasts them. Meanwhile, an attacker generates a random number of events, and then signs each event with multiple pseudonyms and broadcasts them. Due to the small number of event types in our simulation, there is a high probability that two or more attackers report the same event, thus creating a colluding scenario. This behavior of attackers is called "semi-collusion", because there are times that they

report events individually. We intentionally create this a behavior to test P^{2}DAP's resilience to colluding attackers. On overhearing vehicles reporting or relaying an event, an RSU waits for a random time between 20 and 25 seconds before trying to detect Sybil attack and forwarding the reports to the DMV, such that it can ensure all the reports on this event are received.

In the following subsections, we will present our results for the following P^{2}DAP variants:

- C-P^{2}DAP - Detects all Sybil attacks.
- E-P^{2}DAP - Detects Sybil attacks that generate false events.
- T-P^{2}DAP - Detects collusions of a threshold number of attackers.
- τ-P^{2}DAP – Detects collusion with a traffic-volume-adaptive threshold number of attackers.

κ-P^{2}DAP only differs from other algorithms in the initialization stage and does not have different behavior when detecting Sybil attacks. Therefore, we do not evaluate its performance in our simulation.

4.4.2 Computational Overhead of Generating Pseudonyms

Assume we have N_V vehicles in total and each vehicle needs N_P pseudonyms. We also assume a hash function that generates evenly distributed hash values. We first calculate an upper bound (defined as N_u) of the expected number of pseudonyms that the DMV must generate for all of the vehicles. We start with $N_P = 1$, which converts the problem into a *coupon collector's problem* (Flajolet et al., 1992). The expected

number of generated pseudonyms is $N_p \equiv N_V \log N_V + \mu N_V + \frac{1}{2} + o(1)$, where $\mu \approx 0.577$. Thus, for $N_P > 1$, $N_u = N_P \times (N_V \log N_V + \mu N_V + \frac{1}{2} + o(1))$. We can easily find from the definition of N_p that its lower bound is given by $O(N_P N_V)$. Therefore, we conclude that, in order to generate a year's pseudonyms, the number of pseudonyms that the DMV must generate is between $O(N_P N_V)$ and $O(N_P N_V \log N_V)$.

We next calculate the cost of generating short-period pseudonyms. In this scenario, the pseudonyms of each vehicle are divided into d equal portions and each portion is hashed with a unique key. Therefore, with each hash key, the DMV must generate N_P/d pseudonyms for each vehicle. In this case, we have

$$N_u = d(\frac{N_P}{d} \times (N_V \log N_V + \mu N_V + \frac{1}{2} + o(1))) \quad (4.2)$$

$$= N_P \times (N_V \log N_V + \mu N_V + \frac{1}{2} + o(1)) \quad (4.3)$$

while the lower bound of the number of pseudonyms is still $O(N_P N_V)$. Therefore, the upper bound and the lower bound of the expected number of generated pseudonyms for short-period keys remains the same. Obviously, in the extreme case where each time interval only has one pseudonym, the expected number of generated pseudonyms will reach the upper-bound N_u.

We then use the simulator to generate the pseudonyms. Because we only need to obtain the number of pseudonyms generated by the DMV, we stopped the simulation right after the initialization stage. The comparison between the theoretical results and the experimental results of generating long-period pseudonyms is shown in Figure 4.2. It can be seen that the experimental results fall between the calculated upper bounds and lower bounds.

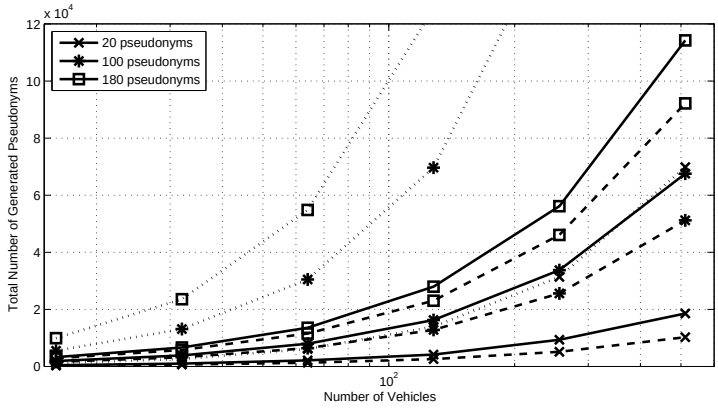


FIGURE 4.2: Computational overhead of the DMV when generating pseudonyms. Solid lines are the simulated results, dotted lines are the theoretically-calculated upper bound values, and dashed lines are the theoretically-calculated lower bound values.

Figure 4.3 also shows that using short-period keys increases the expected number of generated pseudonyms. When dividing pseudonyms into 50 short periods, the number of generated pseudonyms almost doubled, but it does not reach the upper bound N_u.

4.4.3 Experimental Results: Privacy

We first give the definition and metric of privacy in our scenario. If an RSU is compromised, the attacker can obtain the coarse-grained hash keys stored in the RSU, thus learning the coarse-grained hash values of all of the pseudonyms. However, because the coarse-grained hash values are shared among multiple vehicles, the knowledge of a vehicle's coarse-grained hash value does not completely compromise its anonymity. Here we are using the k-anonymity model in (Sweeney, 2002) to evaluate privacy; in order to avoid confusing k in the k-anonymity with our keys, we rename the model

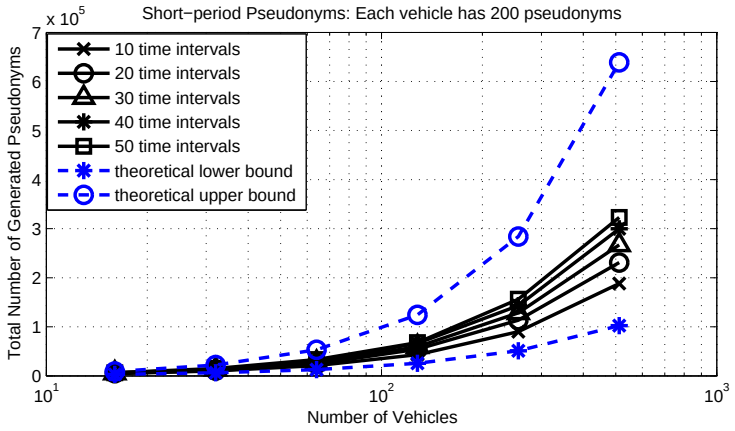


FIGURE 4.3: Computational overhead generating short-period pseudonyms. Each vehicle has a total of 200 pseudonyms.

of privacy as N-anonymity and apply its definition to vehicular networks:

Given a set of vehicles $\{V_i\}_{1 \leqslant i \leqslant N_V}$, a set of attribute values A, and a one-way attribute function $F : \{V_i\} \rightarrow A$, the vehicle set is said to achieve N-anonymity if and only if, for each attribute value $a \in F(\{V_i\})$, there are at least N occurrences of a in $F(\{V_i\})$, where N_V is the number of vehicles.

From this definition, we see that the value of "N" for N-anonymity of the vehicles from an RSU equals the number of fine-grained groups in each coarse-grained group. Therefore, we conclude that the anonymity of vehicles in the case of RSU compromise is 2^{n_f}, where n_f is the number of bits in the fine-grained hash value. In other words, the anonymity is $M/2^{n_c}$, where n_c is the number of bits in the coarse-grained hash value. To study the privacy of vehicles in a subset of all of the vehicles, we generate pseudonyms for 256 vehicles and randomly pick a subset of vehicles to examine their anonymity. The results are shown in Figure 4.4, from which we see that the anonymity of the vehicles quickly converges to 0 when the number of bits of

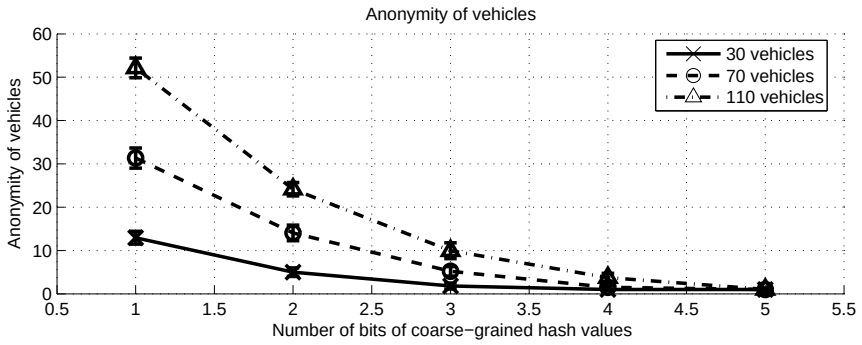


FIGURE 4.4: Anonymity of a subset of all vehicles.

coarse-grained hash values increases to 5. For more vehicles, we expect that a longer coarse-grained hash value will be required to reduce the anonymity. For 2^{24} (more than a million) vehicles, we expect a 20-bit coarse-grained hash value to produce an anonymity of 0.

4.4.4 Experimental Results: Communication Overhead

Overhead on the RSUs

Figure 4.5 shows the number of packets processed by an RSU, while every packet contains a report from a vehicle signed by a pseudonym. Considering that correlation exists between the RSU communication overheads captured at adjacent time points (because we expect similar traffic volumes at time points close to each other), we calculate the 99 % confidence interval of the communication overhead with the algorithm that does estimation with dependent samples introduced in (Trivedi, 2001). In the later observations of DMV communication overhead and detection latency, 99% confidence intervals are calculated with the same algorithm, because we also expect correlation exists between the DMV communication overheads at adjacent

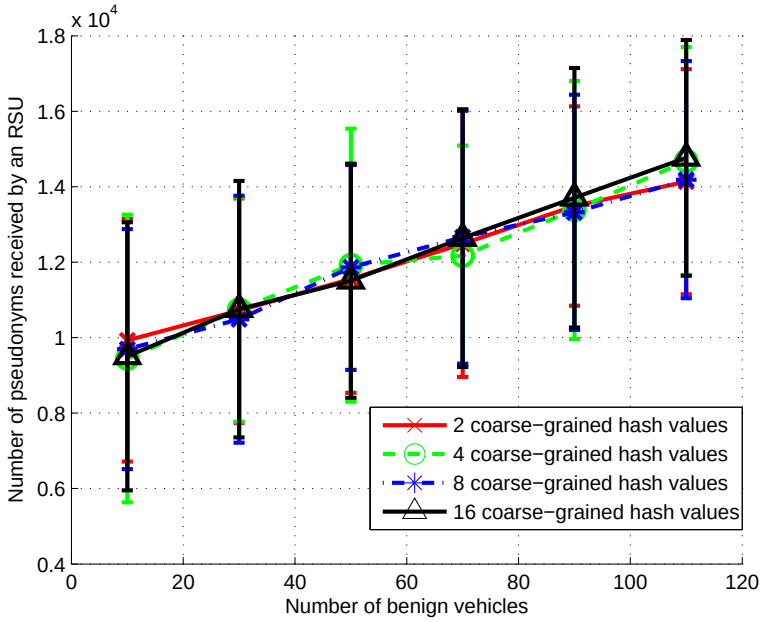
timepoints, and between the detection latency of two consecutively-detected attackers. It is obvious from the figure that the number of packets received by an RSU increases as either the number of attackers or the number of benign vehicles increases. We include these results here for a later comparison of the overhead on the DMV, thus showing the reduction of overhead with the introduction of P^{2}DAP.

Overhead on the DMV

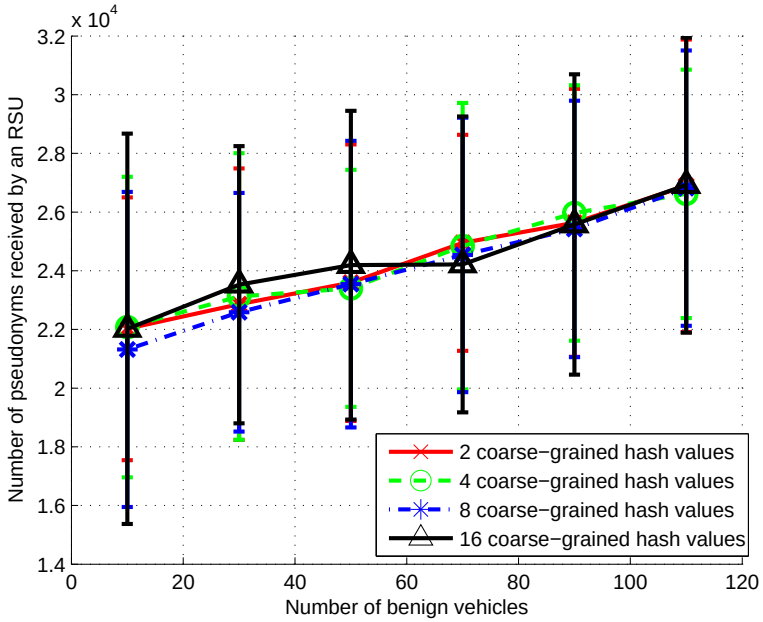
We next examine the number of pseudonyms sent to the DMV when an RSU detects suspicious activities and reports them to the DMV. This metric is indicative of the communication overhead over the backhaul network connecting the RSU and the DMV. Moreover, the number of pseudonyms forwarded by the RSU dictates the computation overhead of the DMV, as the latter must process each packet to detect/confirm a Sybil attack.

- *Overhead on the DMV: C-P^{2}DAP*

  The results of C-P^{2}DAP are presented in Figure 4.6. From the figure, we observe an increase in the number of the forwarded pseudonyms as the number of the benign vehicles increases. This result can be explained as follows: when there are more benign vehicles, an RSU is more likely to send false alarms to the DMV, thus resulting in a larger communication overhead. On the other hand, the total number of pseudonyms forwarded to the DMV always keeps to be the same for increasing number of coarse-grained hash values. We can make the following conclusion from the above observation: in C-P^{2}DAP, when the number of reporting vehicles are much greater than the number of coarse-grained hash values, and the vehicles are evenly distributed, the number of coarse-grained



(a) 3 attackers



(b) 7 attackers

FIGURE 4.5: Number of packets from vehicles to an RSU: for P^{2}DAP.

hash values has little impact on the DMV's communication overhead.

From Figure 4.6, we see that the communication overhead of the DMV is large, thus creating a huge computation overhead for the DMV. Moreover, we can see the communication overhead of the DMV grows linearly over the number of benign vehicles. Therefore, we conclude that, though it is able to detect all the malicious behaviors, C-P^{2}DAP is not scalable to large number of vehicles.

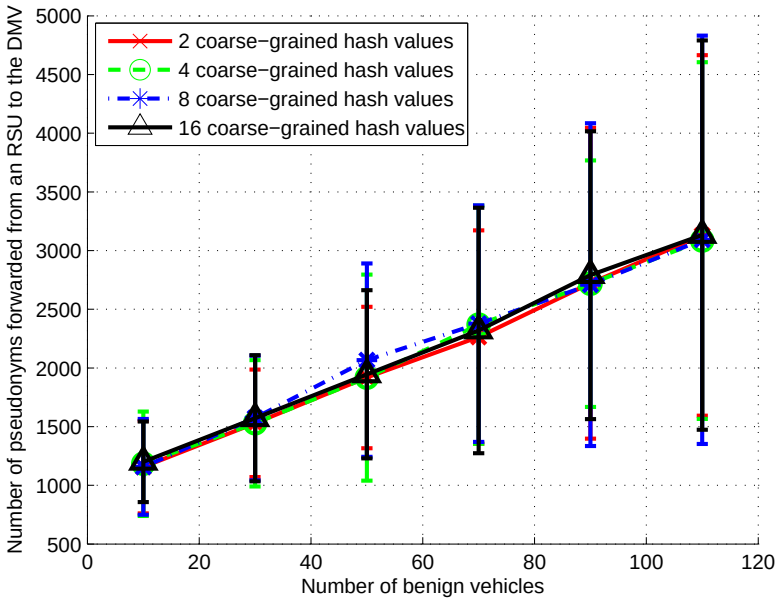
- *Overhead on the DMV: E-P^{2}DAP*

  We next show the relationship between the number of pseudonyms from an RSU and the number of benign vehicles in E-P^{2}DAP in Figure 4.7. By comparing the results in Figure 4.7 and Figure 4.6, we find that the number of pseudonyms received by the DMV is much less when using E-P^{2}DAP, which means that E-P^{2}DAP can efficiently distribute the job of detecting Sybil attack to the RSUs. Moreover, from Figure 4.7, we observe that the communication overhead of the DMV remains almost unchanged when the number of benign vehicles increases. We conclude from these observations that E-P^{2}DAP is scalable to a large number of benign vehicles.

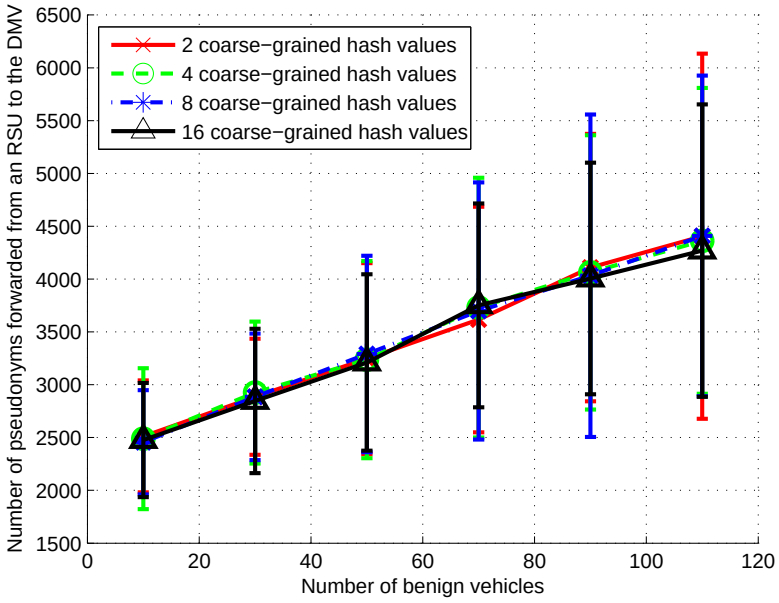
- *Overhead on the DMV: T-P^{2}DAP*

  We then examine the DMV overhead for T-P^{2}DAP. Figure 4.8 shows the communication overhead of the DMV for T-P^{2}DAP. We observe an increase in communication overhead when the value of τ increases or when the number of attackers increases. However, the overhead is still lower than C-P^{2}DAP.

- *Discussion on the Three Schemes*

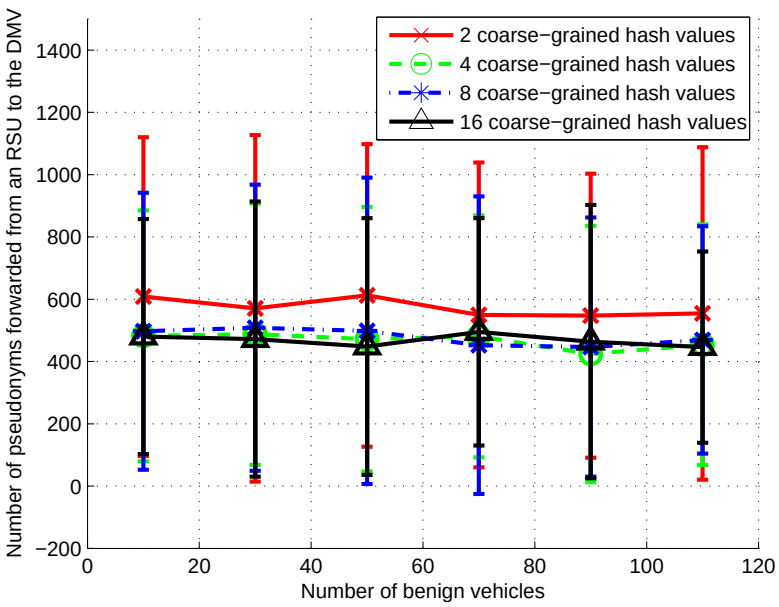


(a) 3 attackers

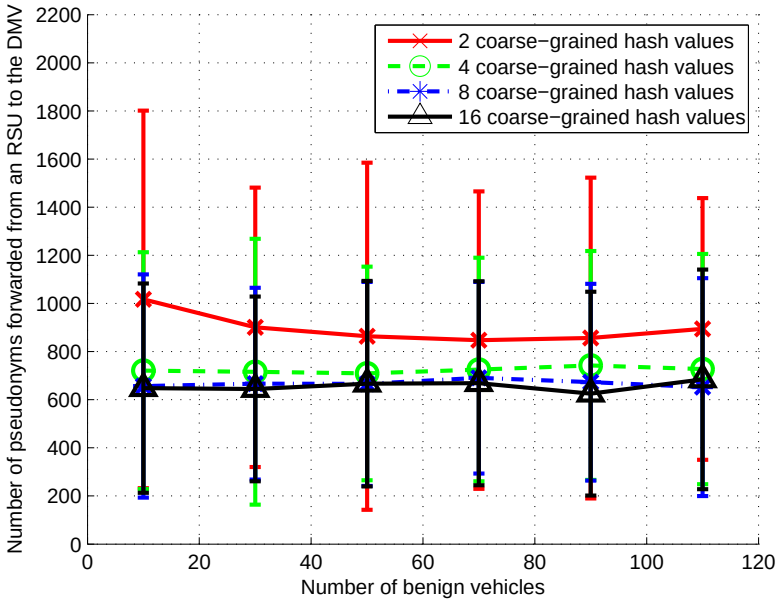


(b) 7 attackers

FIGURE 4.6: Number of pseudonyms from an RSU to the DMV: C-P^{2}DAP.

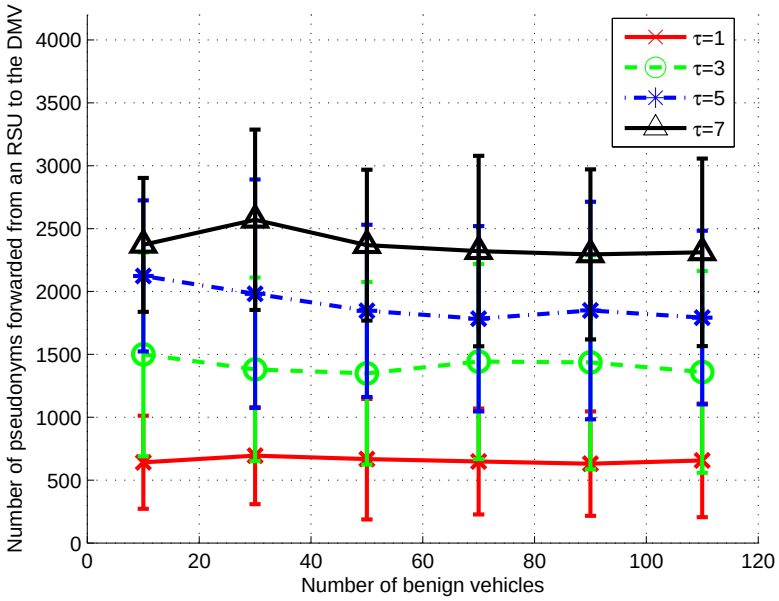


(a) 3 attackers

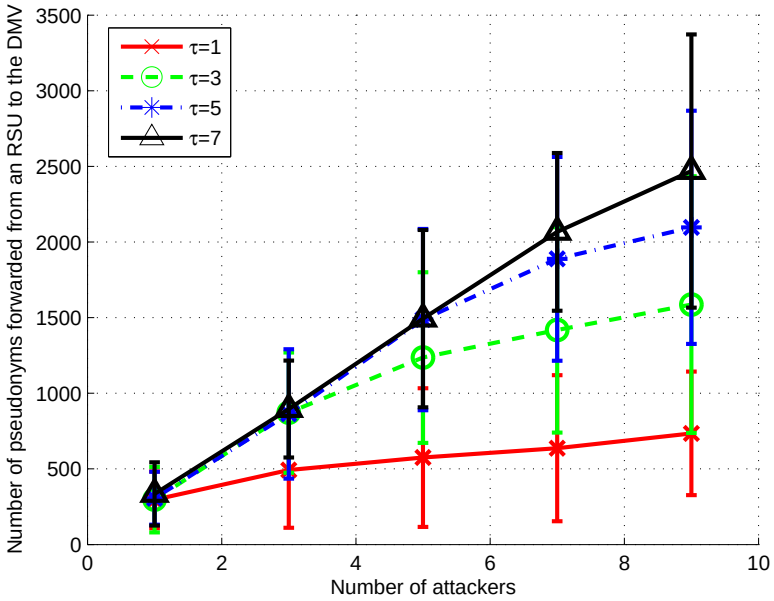


(b) 7 attackers

FIGURE 4.7: Number of pseudonyms from an RSU to the DMV: E-P^{2}DAP.



(a) 7 attackers



(b) 90 benign vehicles

FIGURE 4.8: Number of pseudonyms from an RSU to the DMV: T-P^{2}DAP with 8 coarse-grained hash values.

As shown above, the C-P^{2}DAP is costly. Compared to C-P^{2}DAP, E-P^{2}DAP has a significantly reduced communication overhead for the DMV. On the other hand, T-P^{2}DAP serves as a compromise between C-P^{2}DAP and E-P^{2}DAP, because it is adaptive and its communication overhead value falls between C-P^{2}DAP and E-P^{2}DAP. One interesting observation from a comparison of Figure 4.6 and Figure 4.8 is that, even when τ is only one less than the number of coarse-grained hash values, the communication overhead of T-P^{2}DAP is much less than C-P^{2}DAP.

4.4.5 Simulation Results: Latency for Detecting Malicious Vehicles

In our simulation, the latency Δt for detecting an attacker is defined as $t_{detect} - t_{attack}$, where t_{detect} is the time when the attacker is detected by the DMV and t_{attack} is the time when the attacker first attacks. In the scenario with "semi-colluding" malicious vehicles, the resilience to the collaborating malicious vehicles from their detection latency at different schemes can be learned.

C-P^{2}DAP

C-P^{2}DAP guarantees that every Sybil attack can be detected; therefore, Δt is expected to have the smallest value with this scheme. As discussed in Section 4.2, an RSU detects suspicious actions/events once during each time interval. Therefore, the earliest time that an attack can caught is in the next time interval of that attack and Δt is expected to be the length of the time interval.

In Figure 4.9, we show Δt for C-P^{2}DAP. We observe that with increasing numbers of benign vehicles or attackers, the average value of Δt always remains around 22

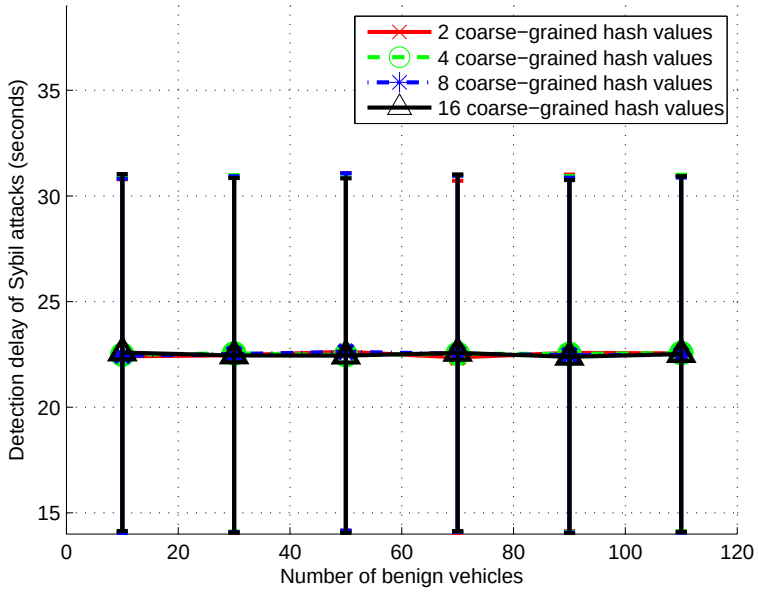
seconds, which is the length of RSU waiting time in the simulation. This observation matches our expectation.

E-P^{2}DAP

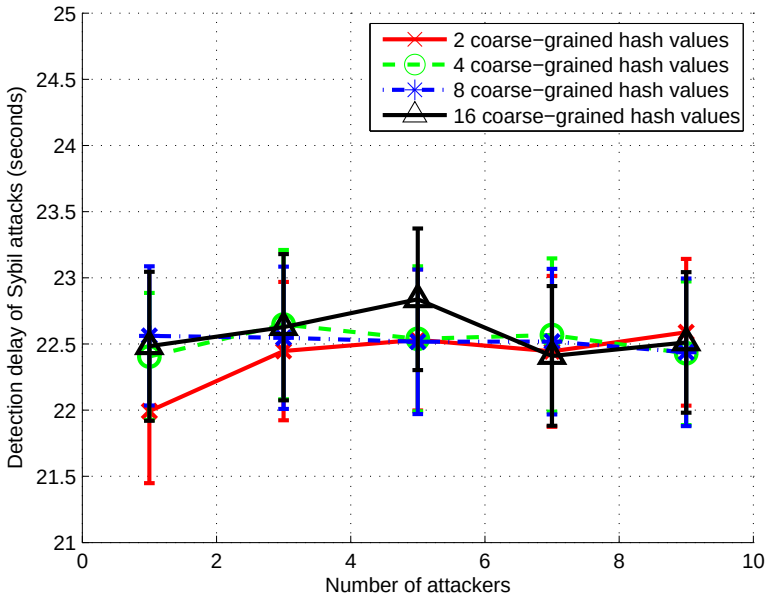
We next examine Δt of E-P^{2}DAP in Figure 4.10. We observe that the number of benign vehicles has little impact on Δt. This is because the probability that all of the vehicles reporting an event have the same coarse-grained hash value is small. On the other hand, we observe an increasing value of Δt as the number of attackers increases, which can be explained as follows.

E-P^{2}DAP is incapable of detecting colluding attackers. However, once an attacker reports an event by itself, it will be detected and then revoked. As more attackers are detected, the probability of attackers colluding decreases and the remaining attackers are more likely to be detected. In this process, E-P^{2}DAP detects attackers sequentially. When there are more attackers, it takes more time for E-P^{2}DAP to observe events signed by only one attacker, thus prolonging Δt.

Another interesting observation is that, the detection delay increases as the number of coarse-grained hash values increases. This observation can be explained as follows: With fewer number of coarse-grained hash values, colluding attackers are more likely to have the same coarse-grained hash value. In this case, RSUs can report more colluding attackers to the DMV, although E-P^{2}DAP is designated for Sybil attacks carried out by only one attacker.

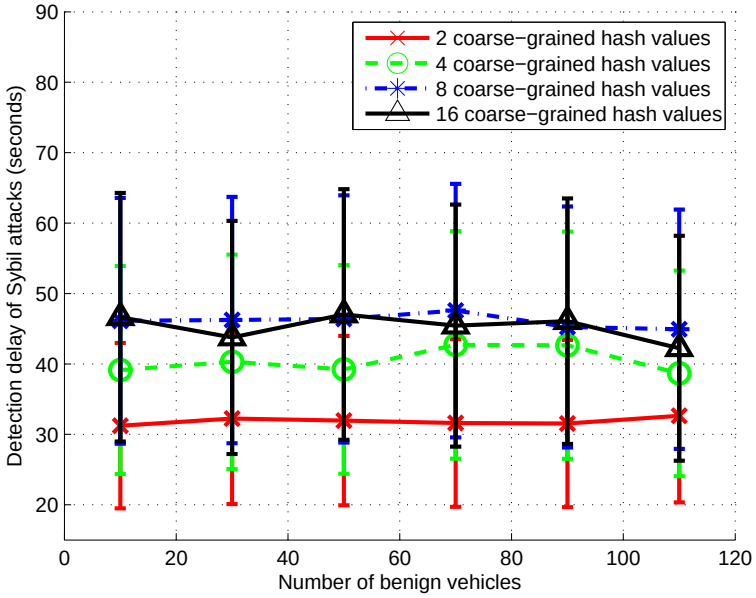


(a) 7 attackers

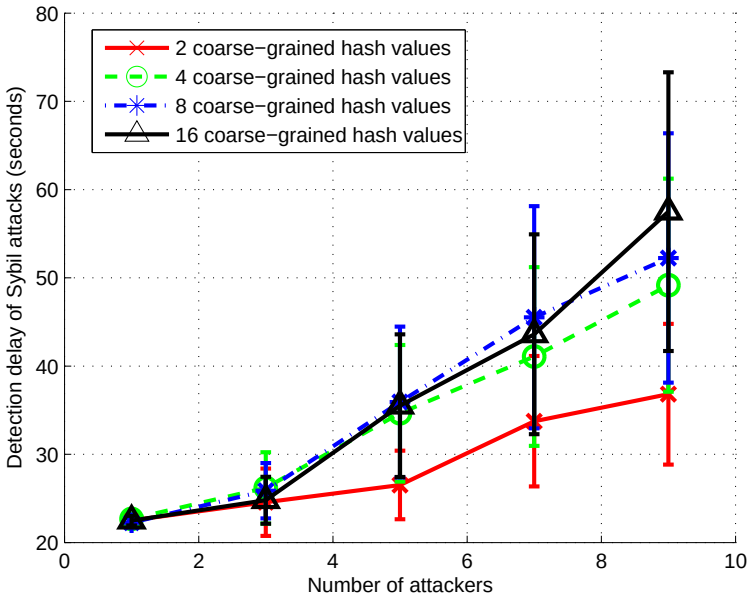


(b) 90 benign vehicles

FIGURE 4.9: Detection latency: C-P^{2}DAP.



(a) 7 attackers



(b) 90 benign vehicles

FIGURE 4.10: Detection latency: E-P^{2}DAP.

T-P²DAP

The detection delay in T-P²DAP is shown in Figure 4.11. From the simulation results, we see when $\tau \geqslant 3$, Δt is reduced to around 22 seconds, which is delay introduced by the RSUs before it forwards the received packets to the DMV. These results indicate that the semi-collusion is resolved by T-P²DAP.

Discussion of the above Three P²DAP Variants

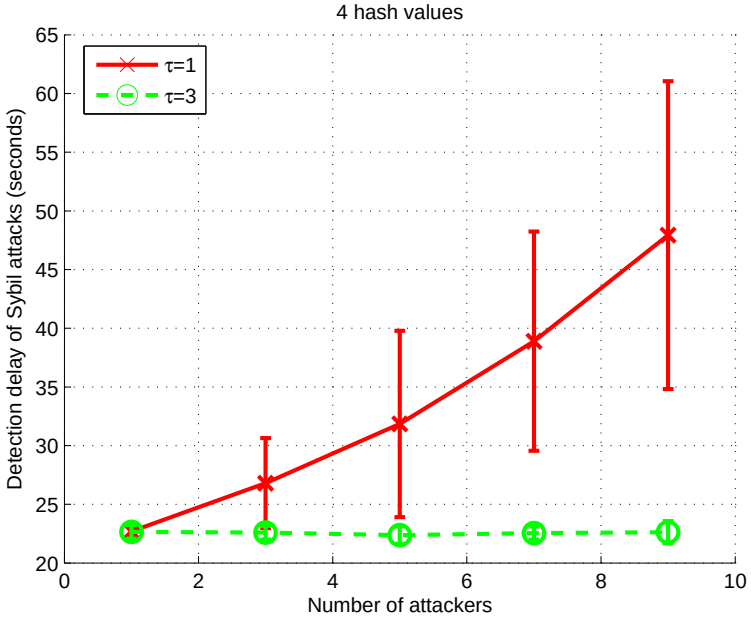
When designing the above three P²DAP variants, according to their resilience to collusion, we would expect Δt to be:

$$\Delta t_{\text{C-P}^2\text{DAP}} < \Delta t_{\text{T-P}^2\text{DAP}} < \Delta t_{\text{E-P}^2\text{DAP}}$$

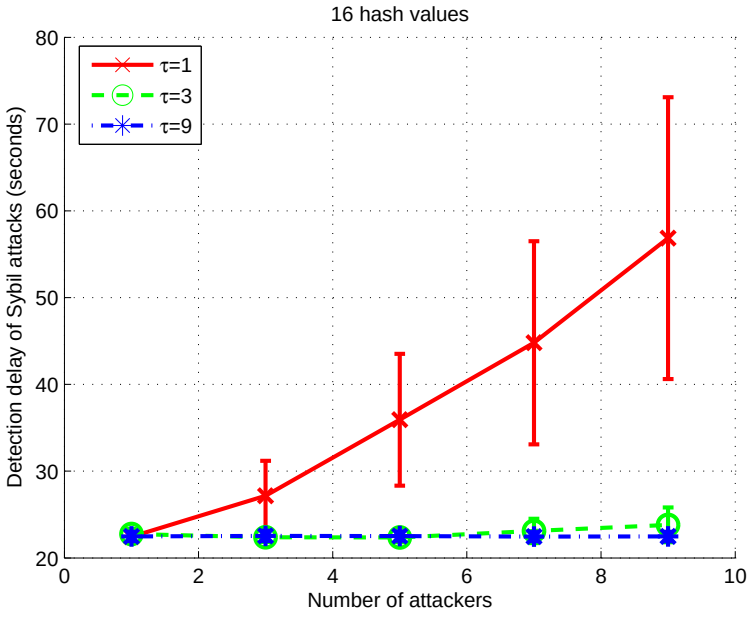
As shown in Figure 4.10, E-P²DAP has the highest Δt, C-P²DAP has the lowest Δt, and T-P²DAP's Δt is between E-P²DAP and C-P²DAP. This observation is expected.

4.4.6 Comparison between T-P²DAP and τ-P²DAP

In sections 4.4.4 and 4.4.5, we compared the performance of C-P²DAP, E-P²DAP, and T-P²DAP. In this section, we specifically study the performance of τ-P²DAP, which by definition is a variation of T-P²DAP with adaptive τ. We compare the performance of τ-P²DAP and T-P²DAP, and show that when τ is adaptive to real-time traffic volume, τ-P²DAP can outperform T-P²DAP in terms of the communication overhead and the detection delay.



(a) 4 coarse-grained hash values



(b) 16 coarse-grained hash values

FIGURE 4.11: Detection latency, T-P^{2}DAP, 90 benign vehicles.

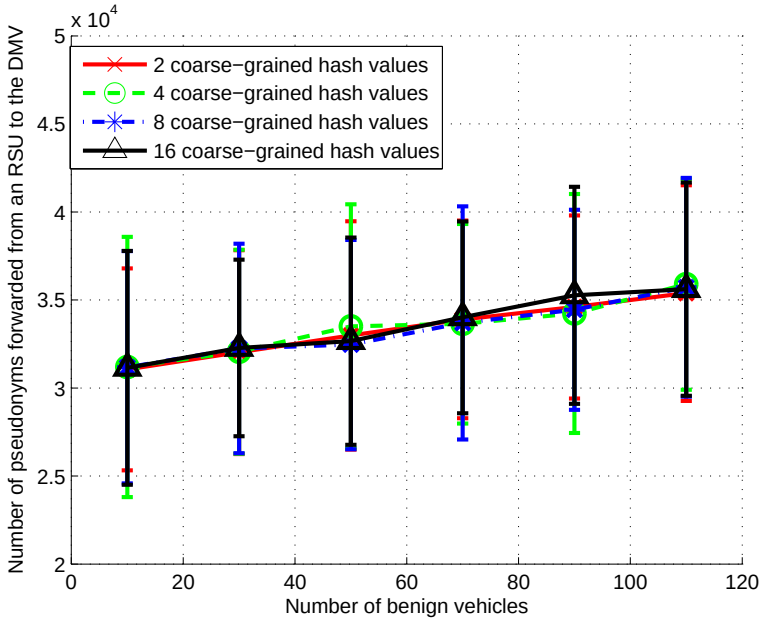


FIGURE 4.12: Communication overhead of an RSU with 10 attackers.

Communication Overhead of the RSU

We first show the communication overhead of the RSU such that we can compare it to the overhead of the DMV. As Figure 4.12 shows, the communication overhead increases as the number of attackers or the number of benign vehicles increases. This result shows similar trend as the results in Section 4.4.4.

Communication Overhead of the DMV

In Figure 4.13, we show the communication overhead of the DMV for T-P^{2}DAP and τ-P^{2}DAP. In both T-P^{2}DAP and τ-P^{2}DAP, we use 4-bit coarse-grained hash values, i.e., there are a total of 16 coarse-grained hash values. We observe increase in the communication overhead for both schemes as the value of τ or α increases. This

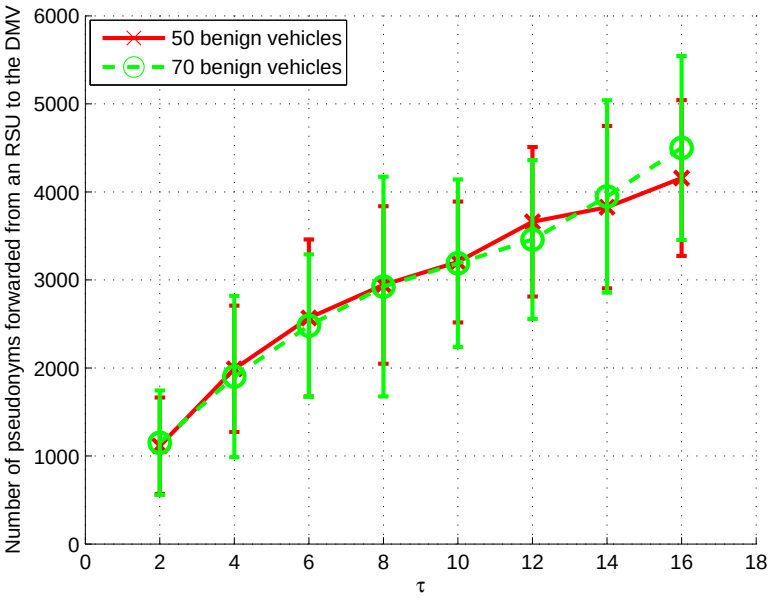
behavior is expected, because when τ or α increase, the RSUs in both schemes are likely to forward more packets to the DMV. On the other hand, we observe that the number of the benign vehicles almost has no impact on the communication overhead. This observation means that both T-P^{2}DAP and τ-P^{2}DAP can efficiently filter out the benign vehicles.

From Figure 4.13(b), we see that when $\alpha = 0.09$, the communication overhead of τ-P^{2}DAP is even less than T-P^{2}DAP with $\tau = 2$.

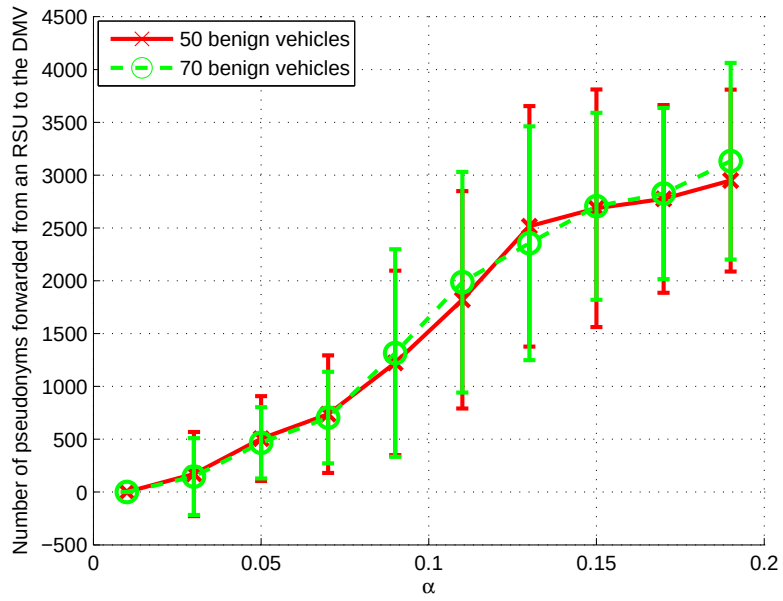
Detection Latency

We next check the value of Δt of the above two P^{2}DAP variants. From Figure 4.14, we observe that Δt of τ-P^{2}DAP quickly drops to around 23 seconds when $\alpha > 0.09$ and then maintains a constant level. On the other hand, T-P^{2}DAP can achieve a similar Δt only when $\tau \geqslant 4$.

When combined with the conclusion from the communication overhead, we conclude that when $0.09 \leqslant \alpha \leqslant 0.13$, τ-P^{2}DAP achieves a good tradeoff between overhead and latency. Specifically, when $\alpha = 0.11$, Δt reaches 22 seconds at a cost of around 1990 packets of communication overhead on the DMV. Such a trade-off cannot be achieved by using a fixed value of τ in T-P^{2}DAP, with the following reason. To achieve the same value of Δt with T-P^{2}DAP, we require $\tau \geqslant 6$; while to achieve the same level of communication overhead, we require $\tau \leqslant 4$. The two requirements of the value of τ cannot be satisfied simultaneously. Also, note that the actual percentages of attackers are 12.5% (for 70 benign vehicles and 10 attackers) and 16.7% (for 50 benign vehicles and 10 attackers). From these observations, these proportions of attackers among all the vehicles are in the "best" range of α. Therefore, we

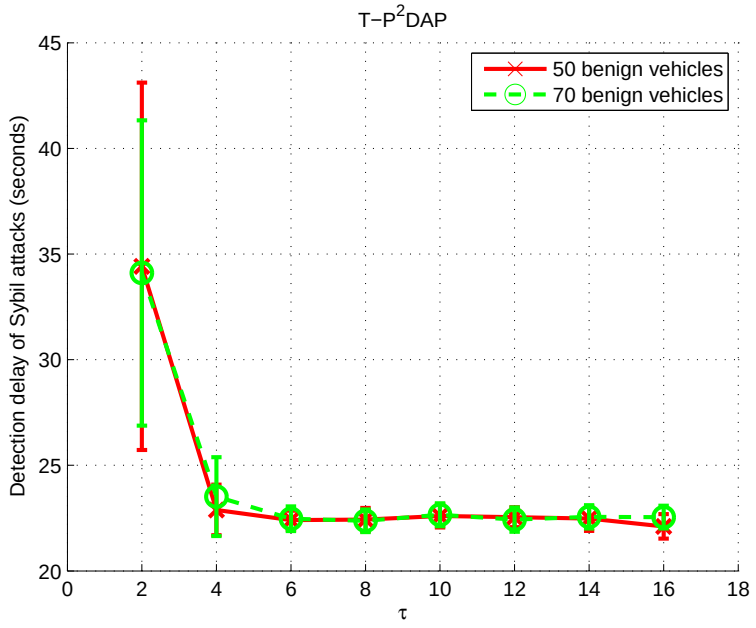


(a) T-P^{2}DAP

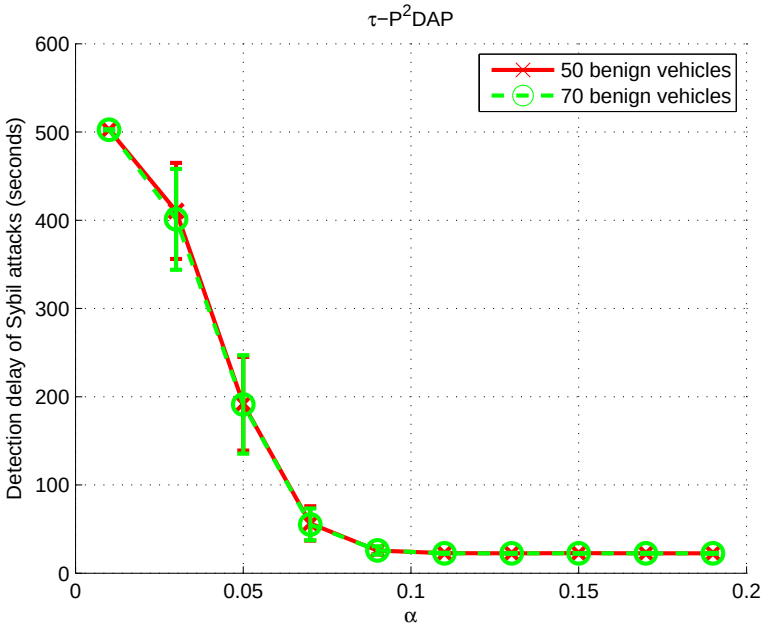


(b) τ-P^{2}DAP

FIGURE 4.13: Communication overhead of the DMV, 10 malicious vehicles.



(a) T-P^{2}DAP



(b) τ-P^{2}DAP

FIGURE 4.14: Average detection latency of 10 malicious vehicles.

conclude that, once a proper estimation of α can be made, τ-P^{2}DAP can achieve a satisfying trade-off between communication overhead and detection latency.

4.5 Conclusions

In this chapter, we propose a new method to detect Sybil attacks in VANETs. The proposed method distributes the computation workload from the DMV to RSUs while releasing only a limited amount of information by using hash collusions. We also discussed select improvements on our scheme. Based on the presented simulation results, we prove that the idea of distributing DMV workload to the RSUs with limited released information is applicable to other VANET security and privacy applications.

5

Predistribution and Local-Collaboration-Based Authentication

In VANETs, RSUs are viewed as a semi-trusted party, and they have direct contact with the CA, overhear broadcasts from vehicles, report nearby traffic and road statuses, and can be used as key distributors in VANET security systems (Parno and Perrig, 2005). On the other hand, because the RSUs are placed along roads, they may be compromised by attackers. In this situation, attackers can use the compromised RSUs to attack the entire VANET by broadcasting bogus messages, collecting information from passing vehicles, or even maliciously revoke keys stored in vehicles. This chapter proposes an authentication scheme for RSUs based on Shamir secret sharing and the watchdog scheme.

The sections of this chapter are listed as follows. Section 5.1 describes the concept of Shamir secret sharing and proposes a new authentication scheme, referred to as Predistribution and Local-Collaboration-Based Authentication (PLCA). The

analytical analysis of security and other performance metrics is presented in Section 5.2. Section 5.3 presents experimental results obtained from Matlab simulations and from real devices. Section 5.4 concludes the chapter and lists future research directions.

5.1 Proposed Method

In this chapter, we propose a PLCA method to handle the key assignment and revocation of RSUs. This is a distributed method that has little impact on the CA or on the backbone network of a VANET. Additionally, the method can revoke an RSU once the RSU is determined to be compromised. In PLCA, when an RSU tries to send any message, it must obtain permission from nearby sensor nodes, which are placed around it and are highly likely to immediately detect any misbehavior. The RSU can send out messages if and only if a majority of sensor nodes trust it. The method outlined above and presented here is based on two key ideas: Shamir secret sharing and neighborhood collaboration.

5.1.1 Shamir Secret Sharing

Shamir secret sharing involves N players and one dealer (Shamir, 1979). The dealer split one secret s_0 into N shares such that any k (where k is some threshold and $k \leqslant N$) shares can be used to reconstruct the secret as follows:

1. The dealer D creates a random polynomial of order $k-1$ that has a constant term s_0. Let this polynomial be given by

$$p(x) = a_{k-1}x_{k-1} + \cdots + a_1x_1 + s_0$$

2. D publicly chooses N random distinct evaluations points x_i, and secretly distributes to the i-th player the share $(x_i; y_i = p(x_i))$. The players can reconstruct the secret as follows:

   (a) Use Lagrange interpolation to find a unique polynomial p of order $k-1$ such that $p(x_i) = y_i$ for $i = 1, \cdots, k$.

   (b) The constant term of p is the secret that needs to be reconstructed.

5.1.2 Predistribution and Local-Collaboration-Based Authentication

We next introduce the proposed PLCA fscheme. Our method is based on the concept of guard nodes, i.e., the placement of several low-cost sensor nodes (referred to as "guards") around an RSU so that they can detect whether the RSU is compromised. On finding that the RSU is compromised, the guards revoke the RSU.

System Assumptions and Attacker Model

In VANET, we assume that the RSU uses elliptic curve cryptography (ECC) (Miller, 1985) to encrypt and sign its messages to both vehicles and the CA and that the guards of an RSU are low-cost sensor nodes whose power supply can be either from nearby power outlets or batteries. One benefit of using ECC is that the private key in the ECC can be any random number. This allows us to easily generate a key chain for the RSUs. For each RSU, there are m guards deployed around it. The guards overhear all messages that the RSU sends and can quickly determine whether they are malicious [1]. The guard sets for different RSUs may overlap if the RSUs are

[1] In practice, this mechanism may not be able to detect all errant behavior and may generate false alarms (Pottie and Kaiser, 2000); for simplicity, we ignore these issues in this thesis.

close to each other. We also assume that the guards are well protected and that the difficulty of compromising any given sensor node is equal to that of compromising an RSU. Each RSU shares a pairwise key KG_j with its j-th guard. These keys ensure the authenticity and confidentiality of messages exchanged between the RSU and the guards. The pairwise key can be either pre-distributed to the RSU and the guards before deployment or generated and distributed by the RSU after deployment. The guards can then cumulatively provide the RSU with the authentication keys needed to send messages if they trust the RSU.

We also assume that an attacker can conduct an active attacks by pretending to be an RSU and broadcasting bogus messages. An attacker can also replay requests to the guards for a key share to perform a DoS attack, as well as replay the key shares or send false key shares from the guards to the RSU such that the RSU is forced to use incorrect shares to construct the authentication key. In addition, an attacker can compromise both an RSU and all of its guards and use these compromised nodes to broadcast false messages or further disrupt other nodes. Additionally, an attacker could perform passive attacks by eavesdropping on the key shares sent from the guards to an RSU.

Approach 1: Basic Method

We first introduce the basic idea used to achieve our goal, i.e., that the guards can revoke the RSU if they judge that the RSU is compromised. In this scheme, each year is divided into N time intervals and the CA must preload N private keys to the guards or the RSUs during an annual registration. Additionally, N corresponding public keys are distributed to the vehicles. For the i-th time interval, an RSU must obtain

the private key K_i to sign the messages it sends. Before the annual registration, CA uses Shamir secret sharing with a threshold k to split every key K_i into m shares $\{k_j\}_{1 \leqslant j \leqslant m}$ and then pre-distributes all of the key shares to the guards. An alternative method of distributing key shares to the RSU guards is to load the keys to the RSU. The RSU then computes the key shares $\{k_{ij}\}$, sends them to its guards immediately after deployment, and finally removes the keys and key shares from its local memory. After deployment and initialization, the j-th guard holds the shares $\{k_{ij}\}_{i \in [1,N]}$ and the RSU has no information regarding the keys that it will use.

The steps followed during broadcast are as follows: At the beginning of the time interval i, each guard j uses the Secure Network Encryption Protocol (SNEP) protocol (Perrig et al., 2001) to send key shares to the RSU, as follows.

$$\begin{aligned} \text{RSU} &\rightarrow \text{Guard}_j : E_{K_{encr}}(N_s | R_s) \\ \text{Guard}_j &\rightarrow \text{RSU} : E_{K_{encr}}(k_{ij} | C), \\ &\quad MAC_{K_{MAC}}(N_s | C | E_{K_{encr}}(k_{ij} | C)) \end{aligned}$$

Both K_{encr} and K_{MAC} are deduced from the pairwise key KG_j, which is shared between the RSU and the j-th guard. N_s is a nonce, and R_s is the request from the RSU. The message from the RSU to the guard is encrypted so that the guard can easily filter out bogus requests. Note also that C is a counter that prevents the replaying of key shares. It can be seen that the response from the j-th guard ensures both the confidentiality and the authenticity of k_{ij}.

After collecting k key shares, the RSU is able to reconstruct the current key K_i, which is then used to compute the message authentication code (MAC) of every sent message. For each broadcast message M in the time interval i, the RSU uses

K_i to compute its MAC value, which is denoted by $MAC_{Ki}(M)$, and then sends $\langle M \| MAC_{K_i}(M) \rangle$.

It can be seen that our basic method is resistant to attacks involving bogus key share requests and bogus key shares; i.e., the attacker cannot perform a DoS attack by sending or replaying requests and key shares. Additionally, once more than $m-k$ guards declare the RSU to be compromised, the RSU can be revoked in the next time interval, thus increasing the resilience to RSU compromise. However, the basic scheme has several disadvantages. First, each guard must store share j of all the keys. Therefore, if the length of each time interval is short, a very large storage overhead is required. Second, the RSU must request key shares in each time interval, which introduces a large communication overhead. A more detailed analysis is provided in Section 5.2.

Approach 2: Short Key Chains Replacing Keys

Approach 1 requires considerable storage and communication overhead. To reduce the overhead, we use Approach 2, which has a reduced security but fewer memory requirements and less communication overhead. This scheme uses multiple short key chains for authentication. Instead of holding the shares of all of the keys, the guards only need to hold the last keys of the key chains, which can be used to generate the whole key by repeatedly computing its hash value.

Now, instead of splitting each key of the entire key chain, only the last key of key chain K_{iL}, where L is the key chain length, is split into m shares $\{k_{iLj}\}_{1 \leqslant j \leqslant m}$ with threshold k and distributed to the m guards of the RSU that uses key chain K_{ij}. In addition, the root of the Merkle tree is pre-distributed to all of the vehicles, and the

key chain parameters and their certificates, which are the siblings of the Merkle tree nodes on a path from a leaf to the root, are pre-distributed to the RSUs.

Before using the key chain K_{ij}, the RSU broadcasts the parameters of K_{ij} with immediate authentication. After that, the guards use SNEP protocol, as in Approach 1, to send seed shares $\{k_{iLj}\}_{1 \leqslant j \leqslant m}$ to the RSU, as follows.

$$\text{Sender} \rightarrow \text{Guard}_j : E_{K_{encr}}(N_s|R_s) \tag{5.1}$$

$$\text{Guard}_j \rightarrow \text{Sender} : E_{K_{encr}}(k_{iLj}|C), \tag{5.2}$$

$$MAC_{K_{MAC}}(Ns|C|E_{K_{encr}}((k_{iLj}|C)) \tag{5.3}$$

After collecting k shares, the RSU reconstructs the seed K_{iL}, and by repeatedly computing the hash values from K_{iL}, the RSU generate a key chain K_{ij} where $K_{ij} = H(K_{i(j+1)})$. Each key in this key chain is then used as a private key for the RSU.

Similar to Approach 1, an attacker cannot carry out DoS attacks by sending or replaying key share requests and key shares. Additionally, if less than k guards trust the RSU, the RSU is revoked. However, when an RSU is found to be compromised, it can only be revoked after the current key chain is completely used, as the RSU already holds all the keys of the current key chain. The benefit of Approach 2 is that the storage and communication overhead required for each guard is simply the shares of all the seeds, which is $\frac{1}{L}$ of that in Approach 1.

5.1.3 Handling Attacks that Compromise Guards

There is a DoS attack that can seriously threaten the security schemes described in Section 5.1.2. Consider the case that the attacker focuses on a single guard. In this case, the attacker can use the pairwise key shared between the guard and the RSU.

Therefore, the attacker can use the guard to send false key shares authenticated with the pairwise key, and the RSU cannot detect the attack. In this case, the RSU will construct an authentication key or seed with an incorrect share, obtain an incorrect key, and thus be effectively disabled.

The compromising of guards may also facilitate the injection of false data. This is because, with Shamir secret sharing, any k shares can be used to reconstruct the secret. Therefore, in order to spoof an RSU and broadcast false data, an attacker does not necessarily need to compromise the RSU; they only need to compromise k guards. In practice, to reduce the cost of deploying guards, it is possible to make the RSU well-protected and difficult to compromise, while the guards are unattended and remain relatively easier to compromise. In this case, it might be easier for an attacker to compromise k guards than to compromise a single RSU.

The above two problems, namely DoS attacks and false data injection, are both based on guard compromise. Therefore, we next discuss some improvements that can increase resilience to these attacks.

Dividing the Key to be Shared

To reduce the likelihood of false data injection attacks, the following simple modification is added to our scheme. For a seed or a key k_i, we randomly generate a key $k1_i$ and compute $k2_i = k1_i \oplus k_i$. The value of $k1_i$ is then predistributed to the RSU while $k2_i$ is further split into m shares using Shamir secret sharing and predistributed to the m guards. During broadcasting, the RSU obtains k_i by XORing $k1_i$ that is held by the RSU itself, with $k2_i$ that is reconstructed from the shares given by the guards. By this modification we make the RSU essential for the authentication of

each broadcast, thus increasing the difficulty of false data injection. Another benefit of dividing the keys is that the encryption of key shares is no longer needed, as an attacker cannot inject false data even if it obtains all the key shares by eavesdropping.

Adding Resilience to Guard Compromise

We now propose two methods that increase the resilience of our schemes to DoS attacks involving the compromise of select guards and the sending of false shares. The first method is to reconstruct the keys from shares using a Guruswami-Sudan decoder (GS decoder) for the Reed-Solomon code [18] instead of reconstruction with Shamir secret sharing using Lagrange interpolation. The GS decoder requires the same exact key shares as in Shamir secret sharing. However, instead of requiring only k shares for interpolation, all of the received shares are required. By using the Reed-Solomon code for reconstruction, we ensure that if $l_1 \leqslant m$ shares are received by the RSU but l_2 shares are polluted, where $l_1 - l_2 \geqslant k$, the key can still be correctly reconstructed. Therefore, this extension makes our method resilient to DoS attacks if up to $\lfloor (m-k)/2 \rfloor$ guards are compromised.

The second extension is that, for each guard, we chain all of the shares that were predistributed to the guard. Assume that the shares predistributed to a guard are denoted by $\{share_j\}_{1 \leqslant j \leqslant L}$. We then append the hash value of the future shares to the previous keys, as shown in Figure 5.1. Now, each message contains one key share and multiple hash values of future messages. The multiple hash values are used to make our scheme resilient to the loss of multiple message. Because all of the key shares are pre-computed, these messages can be pre-computed and easily distributed to the guards. Finally, the hash value of $Packet_1$ is pre-distributed to the RSU.

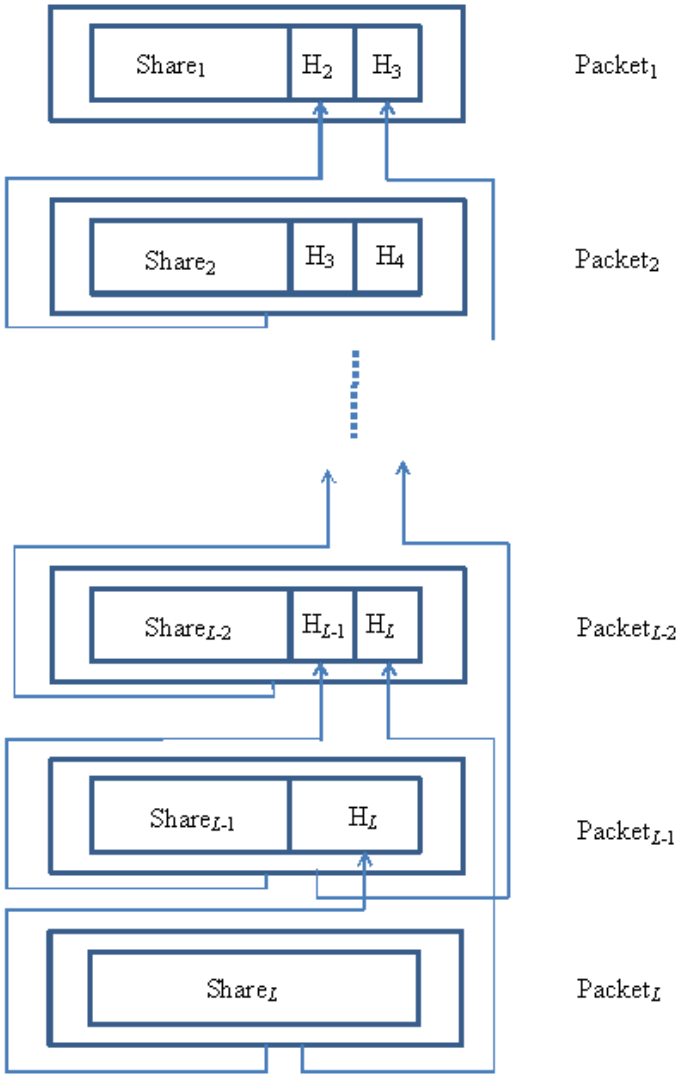


FIGURE 5.1: Chaining the key shares.

After the guards' key shares are chained, the RSU can easily authenticate a message from a guard by computing and verifying its hash value. Therefore, all of the polluted and replayed key shares are filtered out immediately. This method therefore makes our approaches resilient to compromise of up to $m - k$ guards. Additionally, it can be seen that when key share chaining and key dividing are combined, the keys can be efficiently authenticated and secured, thus removing the need for SPIN.

Hiding Guards behind Multi-hops

One-hop guards are easy to compromise, as an attacker only needs to compromise all of the neighbors of an RSU to use the RSU to broadcast false messages. Therefore, the placement of the guards several hops away from the RSU can increase the difficulty of compromising all the guards.

We first consider the case of two-hop guards. In this case, the guards are divided into two classes: watching guards (denoted by $\{W_i\}_{1 \leqslant i \leqslant m}$) and keying guards (denoted by $\{G_i\}_{1 \leqslant i \leqslant m}$). The watching guards are one-hop neighbors of an RSU and can watch the RSU, forward the RSU's request for key shares, and forward key shares to the RSU. The keying guards, which are one-hop or two-hop neighbors, hold the key shares. The number of watching guards is equal to to the number of keying guards, and a watching guard can also serve as a keying guard.

An RSU shares a pairwise key KG_i with each keying guard G_i, and the i-th watching guard W_i forwards requests to the i-th keying guard G_i. Note that G_i and W_i can be the same node. W_i watches the RSU, while G_i watches both W_i and the RSU and also holds the key shares. If W_i is declared to be compromised, G_i does not send the key share to W_i.

When the RSU needs key shares, it sends the same requests as in Section 5.1.2 to W_i and W_i decides whether to forward these requests to G_i. Next, G_i decides whether to send the key shares back to the RSU.

The two-hop scheme can be easily extended to longer hops. By hiding the guards away from the RSU, an attacker must exert more effort to compromise all of the guards. However, the communication overhead is also increased, as it requires more communication to send key shares from a keying guard to an RSU.

5.2 Performance Analysis: Analytical Results

5.2.1 Security and Robustness

PLCA increases resiliency to node compromise by distributing private keys of an RSU to m nearby guards. With the use of guards, the difficulty for an attacker to compromise an RSU node increases $k + 1$ times over the original scheme. The robustness of PLCA depends on the value of $m - k$; i.e., the guards can continue to operate until $m - k + 1$ guards become unresponsive or malicious.

Because the attacker can potentially determine the RSU and the guards by carefully observing traffic patterns, the average time of compromising an RSU and using it to inject false data is $t_{cs} + kt_g$, where t_{cs} is the time needed to compromise an RSU and t_g is the time needed to compromise a guard. We assume that an attacker compromises only one node at a time; hence, the time for compromising the entire system is the sum of the individual times.

We next evaluate the computation/communication overhead of PLCA. The following assumptions are made:

- The size of each message is S_M, the size of the key is S_K, the size of each key

share is S_S, the size of MAC is S_{MAC}, the size of nonce is S_N, the size of counter is S_C, and the size of the request is S_R. On average, there are l messages sent during each time interval, there are L keys per key chain, and there are a total of N/L key chains.

- Each authentication uses n_a cycles, each hash function in the key chain uses n_m cycles, generating each key uses n_h cycles, each key split uses $n_s mk$ cycles (where m is number of guards and k is the threshold), and each key reconstruction uses n_{rk} cycles. The key shares must be sent to the RSU securely and with a signature, both encryption and authentication are done with the same pairwise key shared between the RSU and the guard. We assume that the encryption/decryption with the pairwise key take n_e and n_d cycles respectively, and the MAC generation takes n_{MAC} cycles.

An RSU is disabled if either it is compromised or at least $m-k+1$ of its guards are compromised.

Computational Overhead of PLCA

PLCA can be implemented using two methods, and each method has different computational requirements. Approach 1 constructs each key before using it; Approach 2 constructs one seed for each key chain, and then uses the seed to build the key chain.

1. Approach 1. During each time interval, the RSU must decrypt the key share and verify the signature attached to the key shares, reconstruct the key and compute the MAC of each message, and the guards must decrypt the request

from the RSU, encrypt the key share and sign it. Therefore, for each time interval, the computation overhead for the RSU is $n_{rk}+ln_a+m(n_e+n_d+n_{MAC})$ cycles. The computation overhead of each guard is $n_e + n_D + n_{MAC}$ cycles.

2. Approach 2. The following must occur for each key chain: the RSU must reconstruct the seed and use the seed to generate a key chain with length L; i.e., there are $L-1$ total hash computations to generate all of the keys in a keychain. Therefore, the computational overhead of the RSU for each time interval is $(n_{rk} + m(n_e + n_d + n_{MAC}))/L + ln_a + n_h$ cycles.

Communication Overhead

In Approach 1, the key shares are sent to the RSU at the beginning of each time interval. In Approach 2, the key shares are sent to the RSU before the key chain is used, i.e., at the beginning of each nL-th interval, where $n = 0, 1, 2\cdots$ and L is the length of keychain.

1. Approach 1. In addition to the communication overhead of messages sent to CA and vehicles, the guards must send key shares to the RSU. If the RSU receives enough key shares, it can broadcast signed messages. Additionally, if SNEP is used to protect the key shares, more overhead is required for the protocol. The communication overhead associated with an RSU for each time interval is

$$m(S_s + S_{MAC} + S_N + S_C + S_R) + lS_{MAC}$$

and the communication overhead associated with each guard for each time

interval is

$$S_s + S_{MAC} + S_N + S_C + S_R$$

2. Approach 2. The key shares are only sent before using each key chain. Therefore, the communication overhead of the RSU is

$$\frac{m(S_s + S_{MAC} + S_N + S_C + S_R)}{L} + lS_{MAC}$$

and the communication overhead of guard is

$$\frac{S_s + S_{MAC} + S_N + S_C + S_R}{L}$$

Storage Overhead

PLCA requires pre-loading enough key shares to the RSUs and guards to last for one year. This requires storage overhead on these nodes. The commonly-used private key size from ECC is 128 bits. If each time interval is an hour, the storage overhead required on an RSU/guard is less than 150KB, which is pretty small. For Approach 2, in which we trade a delay in the removal of a compromised RSU for less communication overhead, the storage overhead can be even smaller.

5.3 Experimental Results

In this section, we present computation and communication overhead observed on real devices and simulation results using Matlab that describe the security of PLCA.

In VANETs, RSUs are usually considered to be semi-trusted with sufficient computation and communication resources, and the computational overhead of the RSU

is trivial. Therefore, in the experiment, we mainly focus on the computational overhead of the guard nodes, which can be inexpensive sensor nodes with a limited power supply and computational power.

A Samsumg Nexus S smartphone is used to implement the SNEP protocol used for key share distribution and exchange. The smartphone is equipped with a 1GHz CPU, 512MB of RAM, and a Wi-Fi 802.11 b/g/n card. The time for both computation and communication are collected from the phone.

In the Matlab simulation, we estimate the time it would take to compromise each node, evaluate the time taken to fully compromise an RSU, and then use the time to compromise the RSU as a metric of security.

5.3.1 Computational Overhead

The communication between an RSU and its guards can use symmetric key cryptography. We assume that SHA-1 is used to compute the MAC of each message and that AES is used for encryption. We also assume that the length of all of the keys (including the keys used for AES/SHA-1/ECC) and message digest (MD) are 128 bits, i.e., 16 bytes.

In the experiment, we implemented the functions for handling the incoming SNEP message and generating the outgoing message on the smartphone. We also added a function to calculate the total time elapsed for both functions. After running 100 rounds of the SNEP message handling/generation, the observed mean time and standard error of computation is shown in Table 5.1.

In the scenario in which the RSU requests a new key at the beginning of every hour, that accumulates to 27.0 seconds of computational time for the entire year; in

Table 5.1: Computational overhead of the guards in PLCA.

Operation	Mean overhead (ms)	Std overhead (ms)
Receive request	0.35	0.13
Send key share w/o keyshare chaining	2.79	1.22
Send key share w/ keyshare chaining	2.96	3.71

the scenario that each key share is sent with two hash values of the upcoming packets, the computational time accumulates to 29.0 seconds. Both results are acceptable.

5.3.2 Communication Overhead

Again, the time elapsed for sending/receiving SNEP messages is calculated on the smartphone, this time using the wifi connection. When implementing SNEP, we assume the protocol is on top of the UDP layer; therefore, it contains the headers of UDP, IP and the 802.11 WLAN header. In this case, the RSU-to-guard message has 90 bytes and the guard-to-RSU message has 106 bytes for the scheme without key share chaining. For key share with chaining, in which each message carries the MD of two previous messages, the two message sizes become 122 and 138 bytes, respectively.

We implemented a TCP client-server app on the smartphone to simulate the process of SNEP message exchange for both schemes. Table 5.2 shows the mean and standard deviation of the round-trip time for both schemes, and Table 5.3 shows the packet size for both schemes.

For an entire year's use, the total round-trip time for each scheme is 89.68 and 105.14 seconds, respectively. The amount of data transmitted over the year in the

Table 5.2: Communication overhead in terms of RTT and packet size for SNEP.

Scheme	Mean RTT (ms)	Std RTT (ms)
w/o keyshare chainning	10.24	2.97
w/ keyshare chaining	12.00	2.06

Table 5.3: Communication overhead in terms of packet size for PLCA.

Scheme	Request size (bytes)	Keyshare packet size (bytes)
w/o keyshare chainning	90	106
w/ keyshare chaining	122	138

two schemes accumulate to 1.7 and 2.2 MB. These communication overheads are also reasonable for a guard.

5.4 Conclusions

We have proposed a watchdog scheme for VANET RSUs that prevents false data injection attacks via RSU compromise. This method is based on a combination of Shamir secret sharing and neighborhood collaboration. Two approaches that rely on the basic scheme are described. We have presented a theoretical analysis as well as simulation results to evaluate this method in terms of the time taken to inject false data. We have also computed and compared the overhead of the proposed approaches.

6

Conclusions and Future Work

With recent development in wireless communication technology, the concept of a networked car is receiving immense attention worldwide. This increasing importance of communications between vehicles has been recognized by major car manufacturers, governmental organizations, and the academic community. Consequently, many systems, standards, and organizations related to VANET have been developed, which include Vehicle Safety Communications Consortium (VSCC) (Ahmed-Zaid et al., 2011) in the USA, Car-to-Car Communications Consortium (C2C-CC) (Proskawetz, 2007) sponsored by the European Union, and the Advanced Safety Vehicle Program (ASV) (Kinshi, 1996) in Japan.

The fast development in VANET brings with it tremendous business opportunities, but it also introduces many research challenges, ranging from data sensing to centralized traffic control. In this dissertation, we walked through the complete data lifetime in VANET, including collection, processing, dissemination, vehicle privacy,

and security of RSUs. The collected data can be used to achieve a good accuracy in vehicle localization. This information can be efficiently disseminated with a trade-off between the delay and the overhead. The privacy-preserving detection of the Sybil attacks and the revocation of RSUs are also shown to achieve good security and privacy in VANET.

6.1 Thesis Contributions

In this dissertation, we proposed a system of VANET data collection and GPS-free localization, a data dissemination mechanism, a privacy-preserving Sybil attack detection scheme during data dissemination, and a guard scheme to quickly detect and revoke compromised RSUs.

Chapter 2 proposed the GPS-free localization in VANET. Different from previous works, the GPS-free localization is completely independent from the GPS signals and uses cellular tower signals for error correction. We had shown that such a predicting system has a relatively smaller error compared to a base-line localization, i.e., the Smartphone built-in coarse-grained localization.

Chapter 3 proposed the data dissemination method that exploits the social movements of drivers. While most previous works handle the irregular and short-time encounters between vehicles that is less than 10 minutes, we were more focused on regular-and-long-time encounters of tens of minutes or even hours between the drivers. The social behavior of vehicles fills the hole of short vehicle-to-vehicle connections, and allows for a large data exchange between the nodes.

Chapter 4 proposed the privacy-preserving Sybil attack detection mechanism. This method does not depend on a specific hardware setup, e.g., directional antennas,

or exact matching message content. It is also tolerant to the RSU compromise or unavailability. RSUs do not proactively disseminate information to vehicles. Instead, RSUs overhear broadcasts from vehicles, make decisions, and then communicate to the CA via a secured link. The scheme achieves good protection of vehicle privacy, as well as resiliency to Sybil attacks. It is also adaptive to different traffic loads on the road.

Chapter 5 proposed the scheme of detecting compromised RSUs and quickly revoking them. It places inexpensive nodes around RSUs and uses a Shamir secret sharing technique to split the keys/certificates of the RSUs to the surrounding nodes. The experiment had shown that the communication overhead and battery usage of the nodes are very low. While the cost of this scheme is low, it can immediately detect and revoke compromised RSUs, such that they cannot escalate the incorrect information to the rest of the VANET.

6.2 Future Work

As a fast emerging area, VANET is imposed on various problems and research opportunities that are different from traditional Wireless LAN or cellular networks. The areas covered in this dissertation are also open to new research challenges, which will be listed in the following sections.

6.2.1 Utilizing On-board Sensors

In recent years, bicycle speedometers with Bluetooth features have appeared on the market (Deleo, 2007). If the same technique is applied to vehicles, it will allow us to directly access all on-board sensor readings from a Smartphone and properly use

them.

In this dissertation, sensors on a Smartphone were used to localize vehicles via a combination of the following two machine learning techniques: (1) an SVM used to predict road status from sensor readings, and (2) an HMM used to predict the location based on the road status. With sensors equipped on vehicles, we can gain even higher accuracy in localization. For example, the speed and wheel speed sensor readings can be used to estimate the driving distance with higher accuracy, and the torque sensors can also detect a vehicle turn with a lower error rate than the orientation and gyroscope sensors on a Smartphone.

Besides localization, real-time access to on-board sensor readings can also help in avoiding collisions and improving driving safety. One example is the parking assistant sensor. Its purpose is to avoid vehicle collisions and scratching when the vehicle backs into a parking space. However, when using it on the road, the sensor readings can also be used to warn the following vehicles to keep a safe distance.

6.2.2 GPS-free Localization on Bicycles

In some developing countries, bicycles are much more commonly used than motorized road vehicles. In the United States, some bicycle-friendly areas also have lots of bicycles on the road. The usage of bicycles motivates the VANET to be composed of both motor vehicles and bicycles.

One possible improvement based on our GPS-free localization algorithm is to install such a system proposed in Chapter 2 on bicycles. Since bicycles share the road with other vehicles, the disseminated information can be helpful to both bicycles and motorized road vehicles. With a Smartphone dock, bicycles can easily

join in VANETs and contribute to the data collection and dissemination process. Considering that bicycles follow the same traffic rules, but proceed at a slower speed than cars, it is expected that the bicycle would benefit from previous sensor readings collected by a vehicle. Moreover, because bicycles travel at slower paces than cars, further improving the accuracy of location prediction also sounds reasonable. In the following scenarios, bicycles are expected achieve better performance than vehicles and pedestrians:

1. On a road where no motor vehicles are allowed, bicycles can act as data mules and carry packets for dissemination. In many parks or remote villages, there are trails that only bicycles and pedestrians are allowed. Bicycles can achieve a shorter data delivery delay in these places than pedestrians.

2. On a road segment with sparse vehicles, i.e., connections between vehicles are few and very unstable, the slow bicycles take a longer time to go through the road, thus having more opportunity to encounter other vehicles and create more connections in the VANET.

One challenge of vehicle-bicycle communication is mostly due to the different viewpoints of the two types of transportation. When comparing to vehicles, bicycles are usually much closer to the curb or road shoulders. In addition, bicycles travel at a much slower speed than motorized road vehicles. In this case, vehicles and bicycles are likely to have different observations of the same road events. For example, traffic congestion on the inner lane of a road may not be noticed by bicycles located near the curb. These differences must be addressed during the data aggregation process in VANET.

6.2.3 Improved Sybil Attack Detection

One issue in the proposed P^{2}DAP scheme is that it requires a large storage overhead for both vehicles and the DMV. The DMV needs to generate pseudonyms that are sufficient for one year's usage for every vehicle, and the vehicles need to carry all the pre-generated pseudonyms. Therefore, one possible improvement is to combine our scheme and the pseudonym-generating algorithms proposed in (Calandriello et al., 2007; Rass et al., 2008), such that instead of loading a full set of pseudonyms to each vehicle, the vehicles only need to load a set of seeds and then generate the pseudonyms by themselves. Using this approach, we will be able to further reduce the computational overhead of the DMV and increase the privacy of vehicles.

Another interesting piece of future work is to develop a machine-learning algorithm to predict the ratio and activities of malicious vehicles distributedly. With a good estimation of the ratio of the attackers, P^{2}DAP is expected to efficiently catch attackers with a small overhead and delay. Furthermore, introducing the regional DMVs in the centralized management of the resources during the detection process will empower the structure, because DMVs are distributed in different areas in real life.

Other future work includes developing a more efficient method for a partial pseudonym distribution, such that vehicles do not have to carry the pseudonyms for one year's usage. With this improvement, we reduce the loss when a vehicle is compromised. We also expect the ideas of distributing the DMV's duty to multiple RSUs would help applications other than Sybil detection.

6.2.4 Improving RSU Protection

In this dissertation, we proposed a single-hop guarding mechanism to guard RSUs. Future work includes the investigation of a multi-hop guards scheme, in which the RSUs and the guards are harder to compromise. We will also study the addition of false data flows, thus making it difficult to figure out the RSUs and the guards according to the traffic pattern. Finally, we will develop techniques that would allow an RSU to be revoked, even if all the surrounding guards are compromised.

Bibliography

Abakar, M., Saeed, R., Hassan, A., Mohammed, O., Khalifa, O., and Islam, S. (2010), "The challenges of wireless internet access in vehicular environments," in *2010 International Conference on Information and Communication Technology for the Muslim World (ICT4M)*, pp. D31–D36.

Ahmed, A., Gruyer, D., and Mammar, S. (2013), "A new robust cooperative-reactive Filer for Vehicle Localization: The Extended Kalman Particle Swarm 'EKPS'," in *IEEE Intelligent Vehicles Symposium*, pp. 195 – 200.

Ahmed-Zaid, F., Bai, F., Bai, S., Basnayake, C., Bellur, B., Brovold, S., Brown, G., Caminiti, L., Cunningham, D., Elzein, H., Hong, K., Ivan, J., Jiang, D., Kenney, J., Krishnan, H., Lovell, J., Maile, M., Masselink, D., McGlohon, E., Mudalige, P., Popovic, Z., Rai, V., Stinnett, J., Tellis, L., Tirey, K., and VanSickle, S. (2011), "Vehicle Safety Communications Project Executive Overview," Tech. rep.

Atsmon, A. and Atsmon, D. (2011), "iOnRoad," smartphone app, www.ionroad.com.

Baggenstoss, P. M. (2001), "A Modified Baum-Welch Algorithm for Hidden Markov Models with Multiple Observation Spaces," *IEEE Transactions on Speech and Audio Processing*, 9, 411–416.

Bardin, N., Shabtai, E., Shinar, A., Shmuelevitz, F. Y., Elish, Y., Eisnor, D.-A., and Keret, S. (2009), "Free Community-based Mapping, Traffic, and Navigation App," smartphone app, http://www.waze.com.

Basu, S., Basu, S., Valladares, C. E., Yeh, H.-C., Su, S.-Y., MacKenzie, E., Sultan, P. J., Aarons, J., Rich, F. J., Doherty, P., Groves, K. M., and Bullett, T. W. (2001), "Ionospheric Effects of Major Magnetic Storms During the International Space Weather Period of September and October 1999: GPS Observations, VHF/UHF scintillations, and in Situ Density Structures at Middle and Equatorial Latitudes," *Journal of Geophysical Research*, 106, 30389 – 30413.

Berard, A. J., Mentzer, J. L., and Nixon, D. C. (1996), "Cellular/GPS system for vehicle tracking," .

Bhoraskar, R., Vankadhara, N., Raman, B., and Kulkarni, P. (2012), "Wolverine: Traffic and Road Condition Estimation using Smartphone Sensors," in *International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1–6.

Bo, C., Li, X.-Y., Jung, T., Mao, X., Tao, Y., and Yao, L. (2013), "SmartLoc: Push the Limit of the Inertial Sensor Based Metropolitan Localization Using Smartphone," in *International Conference on Mobile Computing and Networking (MobiCom)*, pp. 195–198.

Brown, A. K. and Sturza, M. A. (1993), "Vehicle Tracking System Employing Global Positioning System (GPS) Satellites," .

Burgess, J., Gallagher, B., Jensen, D., and Levine, B. (2006), "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks," in *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006).*, pp. 1–11.

Calandriello, G., Papadimitratos, P., Lioy, A., and Hubaux, J.-P. (2007), "Efficient and Robust Pseudonymous Authentication in VANET," in *ACM International Workshop on VehiculAr Inter-NETworking (VANET).*

Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., and Scott, J. (2006), "Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms," in *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006).*, pp. 1–13.

Cheng, L., Henty, B., Stancil, D., Bai, F., and Mudalige, P. (2007), "Mobile Vehicle-to-Vehicle Narrow-Band Channel Measurement and Characterization of the 5.9 GHz Dedicated Short Range Communication (DSRC) Frequency Band," *IEEE Journal on Selected Areas in Communications*, 25, 1501–1516.

Cheng, Y.-C. (2008), "CRAWDAD data set ucsd/cse (v. 2008-08-25)," Downloaded from http://crawdad.org/ucsd/cse/.

Choi, J. Y., Jakobsson, M., and Wetzel, S. (2005), "Balancing auditability and privacy in vehicular networks," in *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks.*

Collins, M. (2002), "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron algorithms," in *ACL-02 Conference on Empirical Methods in Natual Language Processing*, pp. 1–8.

Conner, M. (2011), "Automobile Sensors May Usher in Self-driving Cars," EDN Network, [posted p26-May-2011].

Cortes, C. and Vapnik, V. (1995), "Support Vector Machine," *Machine Learning*, 20, 273–297.

Daly, E. M. and Haahr, M. (2007), "Social Network Analysis for Routing in Disconnected Delay-tolerant MANETs," in *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '07, pp. 32–40, New York, NY, USA, ACM.

Deleo, J. L. (2007), "Bike Tech: 10 Gadgets for the Ultimate Ride," PC Magazine, http://www.pcmag.com/article2/0,2817,2163475,00.asp.

Ding, C., He, X., Zha, H., Gu, M., and Simon, H. (2001), "A min-max cut algorithm for graph partitioning and data clustering," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pp. 107–114.

Eagle, N. and Pentland, A. S. (2005), "CRAWDAD data set mit/reality (v. 2005-07-01)," Downloaded from http://crawdad.org/mit/reality/.

Elgamal, T. (1985), "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, 31, 469–472.

Flajolet, P., Gardy, D., and Thimonier, L. (1992), "Birthday Paradox, Coupon Collectors, Caching Algorithms and Self-organized Search," *Discrete Applied Mathematics*, 39, 207 – 229.

Franz, W., Wagner, C., Maih ofer, C., and Hartenstein, H. (2004), "FleetNet: Communication Platform for Vehicular Ad Hoc Networks," in *First ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, pp. 259 – 263.

Fujimoto, N., Moriya, M., Ishizuka, A., and Goto, M. (2002), "Intra- and Inter-Vehicle Communication Network Using Low-Cost POF Links," *IEICE Transactions on Information and Systems*, E85-D, 1839–1850.

Gammeter, S., Gassmann, A., Bossard, L., Quack, T., and Van Gool, L. (2010), "Server-side object recognition and client-side object tracking for mobile augmented reality," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 1–8.

Ganapathiraju, A., Hamaker, J., and Picone, J. (2000), "Hybrid SVM/HMM architectures for speech recognition," in *INTERSPEECH*, pp. 504 – 507.

Gao, W., Li, Q., Zhao, B., and Cao, G. (2009), "Multicasting in Delay Tolerant Networks: A Social Network Perspective," in *MobiHoc*.

G.D., J. F. (1973), "The Viterbi Algorithm," *Proceedings of the IEEE*, 61, 268–278.

Golle, P., Greene, D., and Staddon, J. (2004), "Detecting and Correcting Malicious Data in VANETS," in *VANET*.

Gonzalez, M. C., Hidalgo, C. A., and Barabasi, A. (2008), "Understanding individual human mobility patterns," *Nature Letters*, 453, 779 – 782.

Grossglauser, M. and Tse, D. (2002), "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Transactions on Networking*, 10, 477–486.

Guha, S. and Khuller, S. (1998), "Approximation Algorithm for Connected Dominating Sets," *Algorithmica*, 20, 374 – 387.

Han, J., Owusu, E., Nguyen, L. T., Perrig, A., and Zhang, J. (2012), "ACComplice: Location Inference using Accelerometers on Smartphones," in *Communication Systems and Networks (COMSNETS)*, pp. 1–9.

Hao, Y., Cheng, Y., and Ren, K. (2008), "Distributed Key Management with Protection Against RSU Compromise in Group Signature Based VANETs," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1–5.

Hao, Y., Cheng, Y., Zhou, C., and Song, W. (2011), "A Distributed Key Management Framework with Cooperative Message Authentication in VANETs," *IEEE Journal on Selected Areas in Communications*, 29, 616–629.

Hui, P., Crowcroft, J., and Yoneki, E. (2008), "Bubble Rap: Social-based Forwarding in Delay Tolerant Networks," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '08, pp. 241–250, New York, NY, USA, ACM.

Hwang, J. Y., Larson, K., Chin, R., and Holtzman, H. (2011), "Expressive Driver-vehicle Interface Design," in *Proceedings of the 2011 Conference on Designing Pleasurable Products and Interfaces*, DPPI '11, pp. 19:1–19:4, New York, NY, USA, ACM.

IEEE (1999), "IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band," *IEEE Std. 802.11a - 1999*.

IEEE (2010), "IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Amendment 6: Wireless Access in Vehicular Environments," *IEEE Std. 802.11p - 2010*, pp. 1 –35.

Jang, I., Choi, W., and Lim, H. (2011), "An Opportunistic Forwarding Protocol with Relay Acknowledgment for Vehicular Ad-hoc Networks," *Wireless Communications and Mobile Computing*, 11, 939 – 953.

Jansons, J. and Barancevs, A. (2012), "Using Wireless Networking for Vehicular Environment: IEEE 802.11a Standard Performance," in *2012 Second International Conference on Digital Information Processing and Communications (ICDIPC)*, pp. 5–9.

Jeong, J., Guo, S., Gu, Y., He, T., and Du, D.-C. (2011), "Trajectory-Based Data Forwarding for Light-Traffic Vehicular Ad-hoc Networks," *IEEE Transactions on Parallel and Distributed Systems*, 22, 743–757.

Jiang, D. and Delgrossi, L. (2008), "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments," in *IEEE Vehicular Technology Conference (VTC Spring 2008)*, pp. 2036–2040.

Jiang, D., Taliwal, V., Meier, A., Holfelder, W., and Herrtwich, R. (2006), "Design of 5.9 ghz dsrc-based vehicular safety communication," *IEEE Wireless Communications*, 13, 36–43.

Kihl, M., Sichitiu, M., Ekeroth, T., and Rozenberg, M. (2007), "Reliable Geographical Multicast Routing in Vehicular Ad-hoc Networks," in *5th International Conference of Wired/Wireless Internet Communications*, vol. 4517 of *0302-9743*, pp. 315–325, Springer Berlin Heidelberg.

Kim, M., Kotz, D., and Kim, S. (2006), "Extracting a Mobility Model from Real User Traces," in *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006).*, pp. 1–13.

Kim, T. H.-J., Studer, A., Dubey, R., Zhang, X., Perrig, A., Bai, F., Bellur, B., and Iyer, A. (2010), "VANET Alert Endorsement Using Multi-source Filters," in *Proceedings of the Seventh ACM International Workshop on VehiculAr InterNETworking*, VANET '10, pp. 51–60, New York, NY, USA, ACM.

Kimura, T. and Murakami, K. (1993), "An In-Vehicle Communications LSI Set for Automotive Electronic Control Systems," *IEICE Transactions on Electron*, E76-C, 1767–1773.

Kinshi, S.-k. (1996), "Advanced Safety Vehicle (ASV) Technology Put to Practical Use," Website at www.nasva.go.jp/mamoru/en/assessment_car/asv.html.

Kira, K. and Rendell, L. A. (1992), "A Practical Approach to Feature Selection," in *International Workshop on Machine Learning*, pp. 249 – 256.

Kosch, T. and Strassberger, M. (2004), "The Role of New Wireless Technologies in Automotive Telematics and Active Safety," in *8th Symposium Mobile Communications in Transportation*.

Kreßel, U. H.-G. (1999), "Pairwise classification and support vector machines," in *Advances in kernel methods*, pp. 255–268, MIT Press.

LeBrun, J., Chuah, C.-N., Ghosal, D., and Zhang, M. (2005), "Knowledge-based Opportunistic Forwarding in Vehicular Wireless Ad Hoc Networks," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 4, pp. 2289–2293 Vol. 4.

Li, J., Jannotti, J., De Couto, D. S. J., Karger, D. R., and Morris, R. (2000), "A Scalable Location Service for Geographic Ad Hoc Routing," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pp. 120–130, New York, NY, USA, ACM.

Li, Y., Lai, T., Liu, M., Sun, M., and J.Yang (2006), "DTGR: Disruption-Tolerant Geographic Routing for Wireless Ad Hoc Networks," *Simulation*, 82, 399 – 411.

Lin, C., Wu, G., Xia, F., and Yao, L. (2013), "Enhancing Efficiency of Node Compromise Attacks in Vehicular Ad-hoc Networks Using Connected Dominating Set," *Mobile Networks and Applications*, 18, 908 – 922.

Lin, Y., Li, B., and Liang, B. (2008), "Efficient Network Coded Data Transmissions in Disruption Tolerant Networks," in *The 27th IEEE Conference on Computer Communications (INFOCOM 2008)*, pp. 2180–2188.

Lu, R., Lin, X., Liang, X., and Shen, X. (2012), "A Dynamic Privacy-Preserving Key Management Scheme for Location-Based Services in VANETs," *IEEE Transactions on Intelligent Transportation Systems*, 13, 127–139.

Ma, X., Zhang, J., Yin, X., and Trivedi, K. (2012), "Design and Analysis of a Robust Broadcast Scheme for VANET Safety-Related Services," *IEEE Transactions on Vehicular Technology*, 61, 46–61.

Marti, S., Guili, T., Lai, K., and Baker, M. (2000), "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," in *ACM Mobile Computing and Networking*, pp. 255 – 265.

Melgard, T., Lachapelle, G., and Gehue, H. (1994), "GPS signal availability in an urban area-receiver performance analysis," in *IEEE Position Location and Navigation Symposium*, pp. 487–493.

Miller, V. S. (1985), "Use of Elliptic Curves in Cryptography," *Advances in Cryptography*, 218, 417 – 426.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008), "Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pp. 323–336, New York, NY, USA, ACM.

Musolesi, M., Hailes, S., and Mascolo, C. (2004), "An Ad hoc Mobility Model Founded on Social Network Theory," in *International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pp. 20 – 24.

Naumov, V. and Gross, T. (2007), "Connectivity-Aware Routing (CAR) in Vehicular Ad-hoc Networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 1919–1927.

Newson, P. and Krumm, J. (2009), "Hidden Markov map matching through noise and sparseness," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 336–343.

of Transportation, N. Y. S. D. (2002), "List of State Routes In Columbia County," website.

Palla, G., Barabasi, A.-L., and Vicsek, T. (2007), "Quantifying social group evolution," *Nature*, 446, 664–667.

Parno, B. and Perrig, A. (2005), "Challenges in Securing Vehicular Networks," *Fourth Workshop on Hot Topics in Networks (HotNets-IV)*.

Perrig, A., Szewczyk, R., Wen, V., Culler, D., and Tygar, J. D. (2001), "SPINS: Security Protocols for Sensor Networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, pp. 189–199, New York, NY, USA, ACM.

Pottie, G. J. and Kaiser, W. J. (2000), "Wireless Integrated Network Sensors," *Communications of the ACM*, 43, 51–58.

Proskawetz, K.-O. (2007), "Car 2 Car Communication Consortium," Website at www.car-2-car.org.

Rabiner, L. R. and Juang, B.-H. (1986), "An introduction to hidden Markov models," *IEEE ASSP Magazine*, 3, 4–16.

Ramanathan, R., Hansen, R., Basu, P., Rosales-Hain, R., and Krishnan, R. (2007), "Prioritized Epidemic Routing for Opportunistic Networks," in *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking*, MobiOpp '07, pp. 62–66, New York, NY, USA, ACM.

Rass, S., Fuchs, S., Schaffer, M., and Kyamakya, K. (2008), "How to Protect Privacy in Floating Car Data Systems," in *ACM International Workshop on VehiculAr Inter-NETworking (VANET)*.

Raya, M. and Hubaux, J.-P. (2005), "The Security of Vehicular Ad-hoc Networks," in *SASN*.

Raya, M. and Hubaux, J.-P. (2007), "Securing Vehicular Ad-hoc Networks," *Journal of Computer Security*, pp. 39 – 68.

Raya, M., Jungels, D., Papadimitratos, P., Aad, I., and pierre Hubaux, J. (2006), "Certificate Revocation in Vehicular Networks," Tech. Rep. LCA-Report-2006-006, Laboratory for computer Communications and Applications (LCA), School of Computer and Communication Sciences, EPFL, Switzerland.

Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S. J., and Chong, S. (2011), "On the Levy-Walk Nature of Human Mobility," *IEEE/ACM Transactions on Networking*, 19, 630–643.

Ros, F., Ruiz, P., and Stojmenovic, I. (2012), "Acknowledgment-Based Broadcast Protocol for Reliable and Efficient Data Dissemination in Vehicular Ad-hoc Networks," *IEEE Transactions on Mobile Computing*, 11, 33–46.

Ruan, N., Nishide, T., and Hori, Y. (2011), "Threshold ElGamal-based key management scheme for distributed RSUs in VANET," in *Mobile and Wireless Networking (iCOST), 2011 International Conference on Selected Topics in*, pp. 133–138.

Ruj, S., Cavenaghi, M., Huang, Z., Nayak, A., and Stojmenovic, I. (2011), "On Data-Centric Misbehavior Detection in VANETs," in *Vehicular Technology Conference (VTC Fall), 2011 IEEE*, pp. 1–5.

Sampigethaya, K., Huang, L., Li, M., Poovendran, R., Matsuura, K., and Sezaki, K. (2005), "CARAVAN: Providing Location Privacy for VANET," in *ESCAR workshop*.

Schölkopf, B. and Smola, A. J. (2002), *Learning with kernels: Support vector machines, regularization, optimization, and beyond*, MIT press.

Shagimuratov, I. I., Zakharenkova, I., Ephishov, I., and Tepenitzina, N. Y. (2007), "Fluctuations Effects of GPS Signals on High Latitude Transionospheric Links," in *European Conference on Antennas and Propagation*.

Shamir, A. (1979), "How to Share a Secret," *Communications of the ACM*, 22, 612–613.

Shannon, R. V., Zeng, F.-G., Kamath, V., Wygonski, J., and Ekelid, M. (1995), "Speech Recognition with Primarily Temporal Cues," *Science*, 270, 303–304.

Song, C., Wu, J., Yang, W.-S., and Liu, M. (2013), "Catching Up with Traffic Lights for Data Delivery in Vehicular Ad-hoc Networks," in *Proceedings of the 2Nd ACM Annual International Workshop on Mission-oriented Wireless Sensor Networking*, MiSeNet '13, pp. 21–26, New York, NY, USA, ACM.

Studer, A., Bai, F., Bellur, B., and Perrig, A. (2009a), "Flexible, extensible, and efficient VANET authentication," *Journal of Communications and Networks*, 11, 574–588.

Studer, A., Shi, E., Bai, F., and Perrig, A. (2009b), "TACKing Together Efficient Authentication, Revocation, and Privacy in VANETs," in *Conference on SEnsor, Mesh and Ad Hoc COmmunications and Networks (SECON)*.

Suh, W., Yun, H., Chon, K. S., and Park, C. H. (2005), "Forecasting Hourly Traffic Volume of Airport Access Road: Case Study of Incheon International Airport," 2005 Annual Transportation Research Boards' (TRB) Meeeting.

Sweeney, L. (2002), "k-Anonymity: A Model for Protecting Privacy," *International Journal on Uncertainty,. Fuzziness and Knowledge-based Systems*, 10, 557–570.

Trivedi, K. S. (ed.) (2001), *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, John Wiley and Sons, 111 River Street, Hoboken, NJ 07030-5774.

Tsugawa, T., Saito, A., Otsuka, Y., and Yamamoto, M. (2003), "Damping of Large-scale Traveling Ionospheric Disturbances Detected with GPS Networks During the Geomagnetic Storm," *Journal of Geophysical Research*, 108, 1 – 15.

Tuduce, C. and Gross, T. (2005), "A mobility model based on WLAN traces and its validation," in *Proceedings of IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, vol. 1, pp. 664–674 vol. 1.

Vahdat, A. and Becker, D. (2000), "Epidemic Routing for Partially-Connected Ad Hoc Networks," Tech. Rep. CS-2000-06, Duke University.

Wang, T., Cardone, G., Corradi, A., Torresani, L., and Campbell, A. T. (2012), "WalkSafe: A Pedestrian Safety App for Mobile Phone Users Who Walk and Talk While Crossing Roads," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, HotMobile '12, pp. 5:1–5:6, New York, NY, USA, ACM.

Weimerskirch, H., Bonadonna, F., Bailleul, F., Mabille, G., Dell'Omo, G., and Lipp, H.-P. (2002), "GPS Tracking of Foraging Albatrosses," *Science*, 295, 1259.

Wisitpongphan, N., Bai, F., Mudalige, P., Sadekar, V., and Tonguz, O. (2007), "Routing in Sparse Vehicular Ad Hoc Wireless Networks," *IEEE Journal on Selected Areas in Communications*, 25, 1538–1556.

Woo, R. and Han, D. (2010), "Performance of Vehicular-to-Infrastructure Communication Systems based on IEEE 802.11a," in *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*, pp. 453–454.

Wu, T. and Ranganathan, A. (2013), "Vehicle Localization using Road Markings," in *IEEE Intelligent Vehicles Symposium*, pp. 1185 – 1190.

Yang, D., Cai, B., and Yuan, Y. (2003), "An improved map-matching algorithm used in vehicle navigation system," in *Intelligent Transportation Systems*, pp. 1246 – 1250.

Yang, H., Shu, J., Meng, X., and Lu, S. (2006), "SCAN: self-organized network-layer security in mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, 24, 261–273.

Yin, X., Mab, X., and Trivedi, K. S. (2014), "MAC and application level performance evaluation of beacon message dissemination in DSRC safety communication," *Performance Evaluation*, 71, 1–24.

Zarki, M. E., Mehrotra, S., Tsudik, G., and Venkatasubramanian, N. (2002), "Security Issues in a Future Vehicular Network," in *EuroWireless*.

Zemen, T., Bernado, L., Czink, N., and Molisch, A. (2012), "Iterative Time-Variant Channel Estimation for 802.11p Using Generalized Discrete Prolate Spheroidal Sequences," *IEEE Transactions on Vehicular Technology*, 61, 1222–1233.

Zhang, C., Lu, R., Lin, X., Ho, P.-H., and Shen, X. (2008), "An Efficient Identity-Based Batch Verification Scheme for Vehicular Sensor Networks," in *The 27th IEEE Conference on Computer Communications (INFOCOM 2008).*, pp. 816–824.

Zhang, X., Kurose, J., Levine, B. N., Towsley, D., and Zhang, H. (2007), "Study of a Bus-based Disruption-tolerant Network: Mobility Modeling and Impact on Routing," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, pp. 195–206, New York, NY, USA, ACM.

Zhao, J. and Cao, G. (2008), "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, 57, 1910–1922.

Zhou, T., Choudhury, R., Ning, P., and Chakrabarty, K. (2011), "P2DAP: Sybil Attacks Detection in Vehicular Ad-hoc Networks," *IEEE Journal on Selected Areas in Communications*, 29, 582–594.

Biography

Tong Zhou, born on Mar 6, 1981 in China, is currently software engineer in NetApp Inc., and a part-time PhD student of ECE department at Duke University. She received her bachelor's degree from University of Science and Technology of China, and her master's degree from University of Massachusetts, Dartmouth in 2003. She will be earning the PhD degree from Duke University in May, 2015. Her research interests are in mobile networks and network security. She has been a PhD student at Duke since 2004.