CHAPTER 1

INTRODUCTION

A vehicular ad hoc network (VANET) is also known as a vehicular sensor network by which driving safety is enhanced through inter-vehicle communications or communications with roadside infrastructure. It is an important element of the Intelligent Transportation Systems (ITSs). In a typical VANET, each vehicle is assumed to have an on-board unit (OBU) and there is a roadside unit (RSU) installed along the roads. A trusted authority (TA) and maybe some other application servers are installed in the backend. The OBUs and RSUs communicate using the Dedicated Short Range Communications (DSRC) protocol over the wireless channel. The RSUs, TA, and the application servers communicate using a secure fixed network (e.g. The Internet). The basic application of a VANET is to allow arbitrary vehicles to broadcast safety messages to other vehicles and nearby RSUs. Other vehicles may adjust their travelling routes and RSU may inform the traffic control center to adjust traffic lights for avoiding possible traffic congestion.

Like other communication networks, security issues have to be well-addressed. For example, the message from an OBU has to be integrity-checked and authenticated before it can be relied on. Otherwise, an attacker can replace the safety message from a vehicle or even impersonate a vehicle to transmit a fake safety message. For example, an attacker may impersonate an ambulance to request other vehicles to give way to it or request nearby RSUs to change traffic lights to green. Besides, privacy is another important issue in recent years. A driver may not want others to know its driving routes by tracing messages sent by its OBU. Thus an anonymous communication protocol is needed. While being anonymous, a vehicle's real identity should be able to be revealed by a trusted party when necessary. For example, the driver who sent

out fake messages causing an accident should not be able to escape by using an anonymous identity. Thus we call this kind of privacy conditional privacy.

1.1 Overview of VANET

It is stated earlier that in VANET, the connectivity is done among vehicle to vehicle and vehicle to road side units (RSU). The Trusted Authority responsible for the network maintenance. The basic tool for communication is the short range radios that are being installed in any of the nodes. Vehicular node has the shortest transmission range. RSU's are spread sporadically or regularly depending on the deployment of the network in any particular region. In real life RSU's are spread sporadically. They act as an intermediary node between the Trusted Authority (TA) and Vehicular Node (VN).

In 1999, the FCC has allocated a frequency band of 5.850- 5.925 GHz in the US specifically for the purpose of vehicular networks. Similar bands exist in Japan and Europe. The emerging de facto standard for VC is the Dedicated Short Range Communications (DSRC). DSRC has a MAC Layer that is either a modified version of 802.11WLAN or the 3G protocol extended for decentralized access. Since the current 802.11 protocol is not suitable for VANET due to the high mobility and highly dynamic topology, a special version of it, called 802.11p is being developed by the IEEE. Also, the current 3G protocol is designed for centralized cellular networks, but in VANET, centralized infrastructure is not always present.

1.2 Motivation

The increasing mobility of people has caused a high cost for societies as a consequence of the increasing number of traffic congestion, fatalities and injuries. Vehicular Ad-Hoc Networks (VANET's) envisage supporting services on Intelligent Transportation Systems (ITSs), as collective monitoring of traffic, collision avoidance, vehicle navigation, control of traffic lights, and traffic congestion management by signalling to drivers.

VANET's comprise vehicles and roadside equipment owns wireless interfaces able to communicate among them with wireless and multi-hop communication. VANET's are prone to interference and propagation issues, as well as different types of attacks and intrusions, that can harm ITS services. These networks are characterized by high mobility nodes, rapidly changing network topology, wireless links subject to interference, fading due to multipath propagation and highly changing network topologies. The absence of central entities increases the complexity of security management operations, particularly, access control, node authentication and cryptographic key distribution, allowing the participation of misbehaving (malicious or selfish) nodes in the network and posing nontrivial challenges to security design.

Further, wireless communication is susceptible to jamming, eavesdropping and interferences making easy to damage information and service security. Our main motivation comes from the need for the intelligent systems in the coming years which enable us to have a safe and secure journey with entertainment.

1.3 Report Organization

This report is organised as follows

Chapter 1 gives the introduction about VANET, overview of VANET, motivation.

Chapter 2 gives the literature survey, in which we discuss the history of vehicular communication, VANET components and features, security issues, Preliminaries and secure communication in VANET.

Chapter 3 explains the concept of secure and efficient communications in VANETs which includes initial handshaking, message signing, batch verification, real identity tracking and revocation, group key generation, group message signing and verification.

Chapter 4 shows the design of the project which includes system model of VANET, network model, security model, UML diagrams.

Chapter 5 gives the implementation of our project in network simulator which includes installing ns2, various modules such as creation of an environment, placing RSU's, generating public and private keys, registration and message broadcasting.

Chapter 6 shows the experimental results and their analysis which includes simulation model, experiments showing number of vehicles versus delay time and number of vehicles versus error rate and number of vehicles versus success rate.

Chapter 7 gives the conclusion about the work done and future work.

Chapter 8 gives the references which have been considered so far in the implementation of our project.

CHAPTER 2

LITERATURE SURVEY

2.1 History of Vehicular Communication

The original motives behind vehicular communications were safety on the road, many lives were lost and much more injuries have been incurred due to vehicle crashes A driver realizing the brake lights of the vehicle in front of him has only a few seconds to respond, and even if he has responded in time vehicle behind him could crash since they are unaware of what is going at the front. This has motivated one of the first applications for vehicular communications, namely cooperative collision warning which uses vehicle to vehicle communication.

Other safety applications soon emerged as well as applications for more efficient use of the transportation network, less congestion and faster and safer routes for drivers. Using only vehicle to vehicle communications these applications are not efficient. Therefore an infrastructure is needed in the form of the RSU. Although safety applications are important for governments to allocate frequencies for vehicular communications, no safety applications are as important for Intelligent Transportation Systems (ITS).

Besides road safety, new applications are proposed for vehicular networks, among these are Electronic Toll Collection (ETC), car to home communications, travel and tourism information distribution, multimedia and game applications just to name a few. However these applications need reliable communication equipment which is capable of achieving high data rates and stable Connectivity between the transmitter and the receiver under high mobility conditions and different surroundings.

2.2 Vehicular Ad-hoc Networks Components & Features

A VANET consists of vehicles and roadside Base stations that exchange primarily safety messages to give drivers the time to react to life-

endangering events. A vehicle in a VANET is equipped with processing, recording and positioning features and is capable of running wireless security protocols.

Fig 2.2 VANET Architecture

The above figure shows the Vehicular Ad hoc Network (VANET) architecture. It has the capability for the mobile nodes or vehicles to communicate with each other using the internet which will be provided by various service providers. Vehicles request the nearest Road Side Unit (RSU) to deliver the message to the target vehicle. This is very useful in mobile communications. Once a vehicle got registered with the nearest RSU, it can access the inter group communication with the vehicles in that particular group. Once the registration i.e. authentication is complete. RSU shares its own private and public key which plays a major role in communication with other mobile nodes in that RSU group. Hence the grouping of vehicles by RSU is completed by key sharing. Any vehicle which has RSU public and private key is treated as genuine and authenticated.

Base stations exchange primarily safety messages to give drivers the time to react to life-endangering events. A vehicle in a VANET is equipped with processing, recording and positioning features and is capable of running wireless security protocols.

Features

Though vehicular ad hoc networks share general features with conventional ad hoc networks, VANET's have individual characteristics that are decisive in the design of the communication system, these include: *(i)* Dynamic topology, *(ii)* Mobility models, *(iii)* Infinite energy supply and (*iv)* Localization functionality.

Applications

VANET's enable vehicle-to-vehicle *(v2v)* and vehicle-to infrastructure *(v2i)* communication, thus communicating nodes are either vehicles or base stations that can exchange information about traffic issues, road conditions and added value information. According to several authors: VANET's` applications are commonly classified as follows:

- Warning: to prevent detected risky situations.
- Traffic management: to inform about traffic events.
- Added value: to provide numerous services (i.e. Internet).

2.3 Security Issues

Recall that a vehicular network consists of on-board units (OBUs) installed on vehicles, road-side units (RSUs) along the roads, and a trusted authority (TA). We focus on the inter-vehicle communications over the wireless channel. We assume that:

1. The TA is always online and trusted by everyone. RSUs and TA communicate through a secure fixed network.

2. The RSUs have higher computational power than OBUs.

3. The RSU-to-vehicle transmission (RVC) range is at least twice of the inter-vehicle communication (IVC) range to ensure that if an RSU receives a message; all vehicles receiving the same message are in the feasible range to receive the notification from the RSU.

4. There exists a conventional public key infrastructure (PKI) for initial handshaking. The public key of the TA PK_{TA} is known by everyone. The public key of vehicle V_iPKV_i is known by the TA. Also any RSU broadcasts its public

key PKR with hello messages periodically to vehicles that are traveling in the RVC range of it. This PK_R is known by all vehicles nearby.

The private keys of TA, Vi and R are SK_{TA}, SKV_i and SK_R respectively and are kept secret by the corresponding party. To increase the security level, the master key that is picked by TA for every vehicle is not preloaded into any hardware on the vehicle like.

5. The real identity of any vehicle is only known by the TA and itself but not by others.

2.3.1 Security Requirement

1. Message integrity and authentication: A vehicle should be able to verify that a message is indeed sent and signed by another vehicle (or a valid group member) without being modified by anyone.

2. Identity privacy preservation: The real identity of a vehicle should not be linked to any message so that other vehicles or even RSUs cannot reveal a vehicle's real identity by analyzing multiple messages sent by it.

3. Traceability and revocability: Although a vehicle's real identity should be hidden from others vehicles, the TA should have the ability to obtain a vehicle's real identity and to revoke it from future usage.

2.4 Preliminaries

Schemes are pairing-based and defined on two cyclic groups with a bilinear mapping. Now briefly introduce what a bilinear map is and will discuss the basics on the bloom filter, which we apply in the RSU notification phase.

2.4.1. Bilinear Maps

Let G be a cyclic additive group and G_T be a cyclic multiplicative group. Both groups G and G_T have the same prime order q. The mapping $\hat{e}:G\times G \rightarrow G_T$ is called a bilinear map if it satisfies the following properties:

(1) Bilinear: for all P, Q, R Є G and for all a, b Є Z, ê (Q, P +r) = ê (*P+R, Q) = ê (P, Q). ê(R,Q). Also ê(aP,bP)= ê(P,bP)^{a}= ê(aP,P)^{b}= ê(P,P)^{ab}.

(2) Non-degenerate: There exists P, Q Є G such that ê (P, Q) ≠ 1_{GT}.

(3) Computable: There exists an efficient algorithm to compute ê (P, Q) for any P, Q Є G.

The bilinear map ê can be constructed on elliptic curves. Each operation for computing ê (P, Q) is a pairing operation. Pairing operation is the most expensive operation in this kind of cryptographic schemes. The fewer the number of pairing operations, the more efficient the scheme is. The groups G and GT are called bilinear groups. The security of our schemes relies on the fact that the discrete logarithm problem (DLP) on bilinear groups is computationally hard, i.e., given the point Q = aP, there exists no efficient algorithm to obtain a by given P and Q. The implication is that we can transfer Q in an open wireless channel without worrying that a (usually some secret) can be known by the attackers.

2.4.2 Bloom Filter

A bloom filter is a method for representing a set A = a1, a2,. . . , an of n elements to support membership queries. The idea is to allocate a vector v with m bits, initially all set to 0, and then choose k independent hash functions, h1, h2, . . . , hk, each with range 1, . . . , m. For each element a Є A, the bits at the positions h1(a), h2(a), . . . , hk (a) in v are set to 1 (a particular bit might be set to 1 multiple times). To answer if a value b is in A, we check the bits at positions h1(b), h2(b), . . . , hk (b). If any of them is 0, then b is definitely not in the set A. Otherwise we conjecture that b is in the set although there is a certain probability that we are wrong (called a false positive). After inserting n keys into the vector with m bits with k hash functions, the probability that a particular bit is still 0 is $(1-(1-1/m)^{kn})^{k} \sim (1-e^{-kn/m})^{k}$. Let $f(k)= (1-e^{-kn/m})^{k}$ and let $g(k)=Inf(k)=kln(1-e^{-kn/m})$. By finding dg/dk and making dg/dk=0, it can be shown that to minimize the probability of having false positives, k should be set to mln2/n.

2.5 Communication in Vehicular Ad hoc Networks

Vehicular Ad hoc Network (VANET) has been attracting more and more attentions from both academics. It is a critical component of the Intelligent Transportation Systems which aims at enhancing driving safety through inter-vehicle communications or communications with roadside infrastructure. In a typical VANET, each vehicle is assumed to have an On-Board unit (OBU) and there is road-side units (RSUs) installed along the roads. A trusted authority (TA) and may be some other application servers are installed in the backbone.

The On-Board Units (OBUs) and Road Side Units (RSUs) communicate using the Dedicated Short Range Communications (DSRCs) protocol over the wireless channel while the RSUs, TA, and the application servers communicate using a fixed network. Based on this each vehicle can periodically broadcast safety information containing its current speed, location, road condition and traffic accident information, etc. every 100-300ms. With the received information, other drivers can make an early response in case of exceptional situations. With multi-hop forwarding, the messages will be either terminated by an RSU or dropped when exceeding their lifetimes.

The RSU may also inform the traffic control centre to adjust traffic lights for avoiding possible traffic congestion. VANET also provides a platform for a group of known vehicles (e.g. police chasing a bank robber) to establish a secure communication channel (group communication).

2.5.1 Communication between Vehicle and RSUs

If a vehicle wants to send and sign a message to an RSU nearby the following verification process will be carried out:

1. Vehicle first generates a pseudo identity. Different pseudo Identities are used for different messages to avoid being traced. To generate a pseudo

identity, vehicle first generates a random Nonce then its pseudo identity generates.

2. Vehicle sends message to RSU nearby.
3. Upon receiving the message, the RSU finds out the vehicle's verification public key and shared secret key by verification.
4. The RSU then decrypts and verifies the message. If they are Valid, the verification is said to be successful.

CHAPTER 3

SECURE AND PRIVACY ENHANCING COMMUNICATIONS SCHEMES FOR VANETs

3.1 Initial Handshaking

This module is executed when a vehicle meets a new RSU. The vehicle authenticates itself with the TA via RSU. TA will then pass information to RSU to allow RSU to verify the vehicle's signature even if it uses pseudo identity to sign the message. Also, RSU will generate a shared secret with the vehicle. If this is the first time the vehicle authenticates itself with the TA, TA will pass the master key s and a shared secret n to the vehicle. This only needs to be

done once in the whole journey. To increase the security level, s is not preloaded into any hardware on the vehicle. For the shared secret with RSU, a new secret is generated every time the vehicle moves into the region of another RSU.

We use the notations ENCz(M), DECz(M) and SIGz(M) to denote encrypting, decrypting and signing, respectively, message M using the key Z from now on. The detailed processes in this module are as follows:

(1) When a vehicle V meets the first RSU R, it signs it's RID and PWD using its private key SKvi. It then encrypts RID, PWD and SIG_{SKVi}(RID,PWD) using the TA's public key PK_{TA} and sends ENC_{PKTA}(RID, PWD, SIG_{SKVi} (RID,PWD)) to the RSU which forwards it to the TA.

(2) The TA decrypts the block and verifies RID, PWD and checks Vi's signature using its public key PK_{Vi}. If they are all valid and if RID is not in its revocation list, it generates a shared secret ti for Vi and computes Vi's ID Verification Public Key as VPKi=ti XOR RID. TA then passes VPKi to the RSU to enable it to verify signatures from Vi even if Vi uses pseudo identity to sign the message. The TA then stores the (RID, ti) pair into its repository and forwards PK and X=ENC_{PKVi}(S,VPKi,SIG_{SKTA}(S,VPKi)) to the RSU, where PK_R and PK_{Vi} are conventional public keys of the RSU and vehicle Vi respectively. Vi know that s and VPKi are really sent by the TA, the TA includes its signature on s and VPKi(SIG_{SKTA}(s,VPKi)) into the encrypted text.

(3) The RSU chooses a random number mi to be the shared secret between itself and vehicle Vi. It stores the (VPKi, m_i) pair into its verification table for later usage. It then sends Y=ENC_{Pkvi} (mi,SIG_{SKR}(m_i)) and X to vehicle Vi. Again to let vehicle Vi know that mi is really sent by the RSU, the RSU signs it. (4)Vehicle Vi decrypts Y to obtain m_i and verifies the RSU's signature on it. Similarly, it decrypts X to obtain s and VPKi and verifies the TA's signature on them. It then computes its shared secret with the TA using t = VPKi XOR RID.

This basically completes the initial handshaking phase. The following shows the procedure when vehicle Vi leaves the range of an RSU and enters the range of another. It includes a simpler authentication process with the TA so that TA can pass the information to the new RSU for verifying V_i's signature and a new shared secret will be generated by this RSU.

(5) Vi generate a random nonce r' and sends $ENC_{PKTA}(RID||r')$ to TA via this new RSU. The random nonce r' avoids V_i from being tracked even the attacker captures a number of these packets as Vi's moving across RSUs. The TA obtains RID by decrypting the block using its private key SK_{TA} and then removing the concatenation r'. This time the TA does not need to verify Vi's PWD any more as it has already done that when Vi first starts up. Instead it directly generates a new t_i and and a new VPK_i for Vi and sends VPKi to the new RSU. The TA then adds the new ti into its repository. Next the new RSU chooses a random number mi to be its shared secret with Vi. After storing (VPKi, mi) into its verification table, RSU sends $Y = ENC_{PKVi}(mi,SIG_{SKR}(mi))$ to Vi which then decrypts it using its conventional secret key. From now on, vehicle Vi starts to use the new shared secret with the new RSU for message signing.

3.2. Message Signing

To sign a message, a vehicle generates a pseudo identity and the corresponding signing key. A different pseudo identity can be used for a different message.

To generate a pseudo identity, Vi first generates a random nonce r. Its pseudo identity Idi contains two parts–IDi1 and IDi2 where IDi1= r Ppub and IDi2=VPKi XOR H(miIDi1). The corresponding signing key is SKi=(Ski1,SKi2) where Ski1=smiIDi1 and Ski2=sH(Idi2).H(.) is a Map point To Point hash function. Then, to sign a message Mi, Vi computes the signature i=SKi1+ h(Mi)SKi2 where h(.) is a one-way hash function such as SHA-1. Vehicle Vi then sends(Idi,Mi,i) to others.

3.3. Batch Verification

This module allows an RSU to verify a batch of signatures using only two pairing operations based on the bilinear property of the bilinear map. We require an RSU to perform batch verification at a frequency higher than that a vehicle broadcasts safety messages so that a vehicle can verify the safety message of another before it broadcasts a more updated one. We first show the verification procedure. Then, we show how to make use of bloom filter to construct a notification message in order to reduce the message overhead.

3.4 Real Identity Tracking and Revocation

To reveal the real identity of the sender of a message, TA is the only authorized party that can perform the tracing. Given vehicle Vi's pseudo identity ID and its shared secret with the connecting RSU mi, TA can search through all the stored (RIDj,tj) pairs from its repository. Vehicle Vi's real identity is the RIDj value from the entry that satisfies the expression IDi2XORtj XORH(miIDi1)=RIDj. No other party can obtain vehicle Vi's real identity since ti is only known by the TA and Vi itself. Upon getting Vi's real identity RIDi, TA can revoke it if necessary. This can be done by simply storing RIDi into a revocation list. Vi can no longer obtain VPKi from it in the future.

3.5 Group Key Generation

This module shows how a group of known vehicles can form a group with any RSU, then they can communicate securely within the group without any further help from RSU to verify these group messages. Assume that vehicles V1,V2, . . . ,Vn have already registered with an RSU and their shared secrets with the RSU are m1,m2, . . . ,mn respectively. Also assume that these vehicles know pseudo identities of one another already or they can know others' pseudo identities by the last message received from one another.

3.5.1. Group Request

Vehicle Vi first sends to the RSU message M_i = {GPREQ,ID_1, . . . ,ID_{i-1}, ID_{i+1}, . . . ,ID_n} and its signature $r_i = SK_{i1} + h(M_i)SK_{i2}$ on it where ID_j is the pseudo identity of V_j. Also SK_{i1} and SK_{i2} are generated using the methods in Message Signing. Note that V_i can be anyone or the leader of the group.

3.5.2. Group Agree

Any vehicle V_j receiving V_i's GPREQ message checks whether its pseudo identity is included in the GPREQ message. If yes, it sends out M_j = {GPAGR,ID_j } and its signature $r_j = SK_{j1} + h(M_j)SK_{j2}$.

3.5.3. Group Batch Verification

The RSU then batch-verifies $r_1, \ldots, r_n$. For any vehicle V_x whose signature is found to be valid, it generates its group public key as $GPK_x = m_xP$. Recall that m_x is the shared secret between the RSU and vehicle V_x . Besides group public keys, the RSU also requests the TA to provide the group of vehicles a common group secret key. Without loss of generality, assume the signatures from $V_1, \ldots, V_x$ are valid. The RSU sends $VPK_1, \ldots, VPK_x$ to the TA which in turn generates a random number rr and computes the group secret key as CGS = s * rr. Next the TA sends ENC_{t1} (CGS) , . . .,ENC_{tx} (CGS) back to the RSU. Recall that ti is the shared secret between V_i and the TA. The RSU then broadcasts Mr = {ID_1 , . . . ; ID_x , GPK_1 . . ., GPK_x , ENC_{t1}(rr) , . . ., ENC_{tx}(rr)} and its signature SIGSKR (M_r) to the vehicles concerned. Note that in case the verification fails due to invalid signatures or vehicles inside the range have same pseudo identity (although the chance is very small), RSU will stop the protocol and the group is required to repeat the protocol again for the sake of security reason.

3.5.4. Group Secret Establishment

Each vehicle in the group stores all the group public keys and the decrypted CGS values. Note that the RSU does not know CGS since the TA encrypts it using its shared secret with each vehicle. Thus vehicles in the group can communicate with others securely from now on.

3.6. Group Message Signing and Verification

Next we look at the pseudo identity generation, message signing and signature verification when group communications take place. When vehicle V_i wants to send a group message M_i, it generates its pseudo identity ID_i and signature r_i in the same way as in Message Signing. However, its secret signing

key is generated as SK_i = (SK_{i1} ,SK_{i2}) where $SK_{i1} = m_i ID_{i1}$ and $SK_{i2} = m_i H(ID_{i2})$. V_i then sends out {ID_i ,$ENC_{CGS}(GPK_i \,||\, ID_i)$,$M_i$,r_i}, where r is the random nonce used to generate its pseudo identity. Note that GPK_i is included so that the receiving vehicle knows which group public key to use for verification. To make it impossible for any vehicle outside the group to trace V_i, GPK_i is first concatenated with its per session pseudo identity and then encrypted using the common group secret CGS. To verify the signature r_i of vehicle V_i on message M_i, the receiving vehicle first decrypts $ENC_{CGS}(GPK_i \,||\, ID_i)$ using CGS. If it finds that GPK_i obtained does not belong to any group member, it simply ignores the message.

CHAPTER 4

System Design

4.1 System Model

The system model explains about network model of system, privacy of system, group definition of vehicles.

4.1.1 Network Model

In this project consider an infrastructure based VANET's, where entities can be classified into three categories: authorities, roadside infrastructure, and nods. Authorities are responsible for key generation and malicious vehicle judgment. Authorities have powerful firewalls and other security protections. Therefore, they have the highest security level. We assume that they cannot be compromised. Roadside infrastructure consists of RSUs deployed at the road sides which are in charge of key management in our framework. They are the access points in our framework. Each access point has a specific range basing on which the group public and private keys are distributed to the vehicles. Traffic lights or road signs can be used as RSUs after renovation. RSUs communicate with authorities through wired network.

We assume a trusted platform module is equipped in each RSU. It can resist software attacks but not sophisticated hardware tampering. The cost of a trusted platform module is only a few tens of dollars which is affordable. RSUs are semi-trust with the medium security level. Nodes are ordinary vehicles on the road that can communicate with each other and RSUs through radio. Nodes have the lowest security level.

4.1.2 Privacy System

In this framework, the communications can be divided into the key distribution phase and the regular broadcast phase. Vehicles get keys dynamically in the key distribution phase and then start to broadcast their

geographic and road condition messages periodically in the regular broadcast phase.

For provision of privacy, we follow certain technique where the vehicle signs the tracing key list hold by the authorities with its private key. RSU provides the vehicle with group private and public keys whenever it comes within the range of that RSU. RSU maintains the tracing key list of all the vehicles that comes within its range. Usually in VANET's RSU can be compromised and handover the key details of one vehicle to a malicious vehicle. In bellow Figure explains about system model briefly

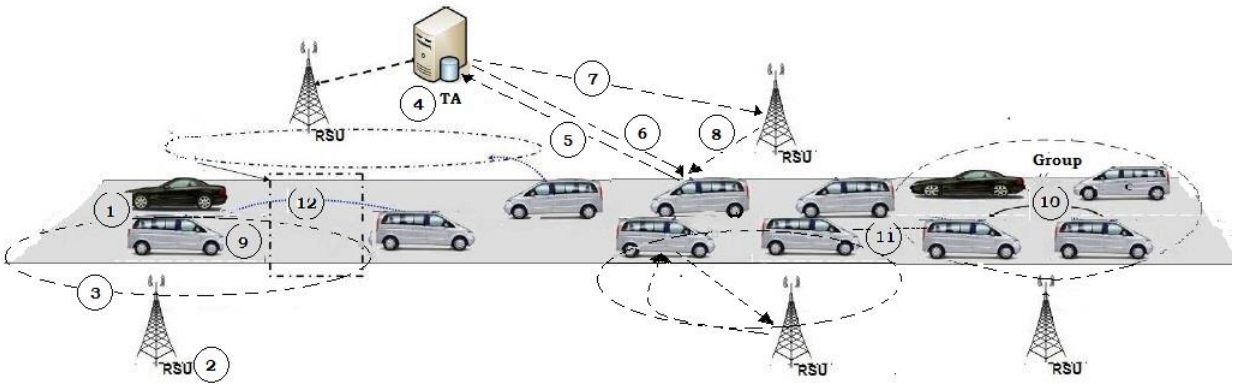

Fig. 4.1 System Model of VANET

Contents of System Model:

In the above figure some numbers are indicated. They are explained as follows:

1 - Road
2 - Road Side Unit
3 - Group of Vehicles in RSU
4 - Trusted Authority
5 - Authenticate with TA
6 - Shared Secret, TA Master Key
7 - Verification Information for Vehicle

8 - Shared Secret Key
9 - On Board Unit
10 - Intra Group Communication
11 - Inter Group Communication
12 - Communication Without a Group

4.1.3 Group Definition

Those vehicles getting keys from the same RSU form a group, as illustrated where the communication range of RSUs shown by a circle. We consider that RSUs are only deployed at entrances/exits of the road segments. In a highway scenario, RSUs are normally far away from each other. In the region out of the RSU coverage, vehicles in the same group can communicate with each other in an ad hoc manner. In a city area, RSUs might overlap with each other. We define that a vehicle is only associated with one RSU at a moment to get the service.

4.2 Security Model

Assume that attackers are inside, rational, active, global and parsimonious. Inside attackers are legitimate members of VANET's. In this project, attackers can be network nodes or road side infrastructure. Rational attackers only attack for their own benefits. They know the security mechanism and they want to attack without being detected. If there is a mechanism that can detect them and the punishment is severe enough, they tend not to attack. Active attackers have the ability to send packets into wireless channels. Global attackers have an unlimited scope which means they can hear any information in the network.

Attackers may have strong transmission power to communicate over long distances. Adversarial parsimony means an attack involving a few malicious nodes is more likely to happen than an attack that requires collusion among a large number of nodes. I assume that the overwhelming majority of vehicles and RSUs are honest which is reasonable in the civilian use system and also

assume vehicles will report to authorities when they find that other vehicles send a false message.

4.3 UML Diagrams

The proposed methodology to detect compromised RSUs and their accomplices which is a brand new security issue induced by the distributed key management framework. A misbehaved RSU will let authorities fail to identify malicious vehicles. This method allows vehicles to be authenticated with their real identifiers under protection and guarantees authorities to find compromised RSUs. Authorities make decisions according to the registration information that vehicles provide. Hereby, the registration procedure is the most important part.

Each vehicle and RSU is preloaded with a global long term public/private key pair with keys are used. The trusted authority (TA) is always online and trusted. RSUs and TA communicate through a secure fixed network. The group public key and group private keys are local, short term keys in the method. They are considered as identifiers of vehicles and RSUs. In this project, binary search and bloom filter techniques in RSU message verification phase to reduce message overhead.

Here different methods are used for the system design. They are:

- Sequence Diagram
- Class Diagram
- Activity Diagram
- Use Case Diagram

All the UML Diagrams are have been developed in Rational Rose.

Sequence Diagram:

The purpose is to show the flow of functionality. In other words, we can call it as mapping process in terms of data transfers from the actor through corresponding objects.

Fig.4.2 Sequence Diagram for Registration Procedure

The vehicle can identify the RSU's legitimacy after it verifies this message because the RSU uses its own private key in the message. The vehicle encrypts its private key and the timestamp by using an authorities' public key. Then, it sends the encryption data with the timestamp and the signature of corresponding information, to the RSU. The encryption of its private key and the timestamp is a commitment. We will use it to detect illegitimate users later. Meanwhile, the signature signed by the vehicle binds vehicle's information and the assigned group private key. Then, the RSU cannot re-map them because the RSU does not have the vehicle's private key. The RSU sends the group private key to the vehicle. The vehicle finishes the registration procedure after it gets a valid group private key. Then, the RSU stores the information .If authority's need the information of a vehicle when there is a dispute, the RSU has to send the vehicle's corresponding information to authorities.

Class Diagram

A class diagram is a diagram that shows a set of classes, interfaces and collaborations and their relationships. Here we represent the implementation classes of vehicles registration and messages broadcasting in VANETs in the form of class diagram. We have represented each class with their attributes and their methods. The sequence of classes represented in the class diagram corresponds to the sequence of classes that are called during execution. The objects present in the above class diagram are vehicles, roadside units, Trusted authority, authenticate, share and request. Attributes and methods corresponding to each object are shown in the above class diagram.

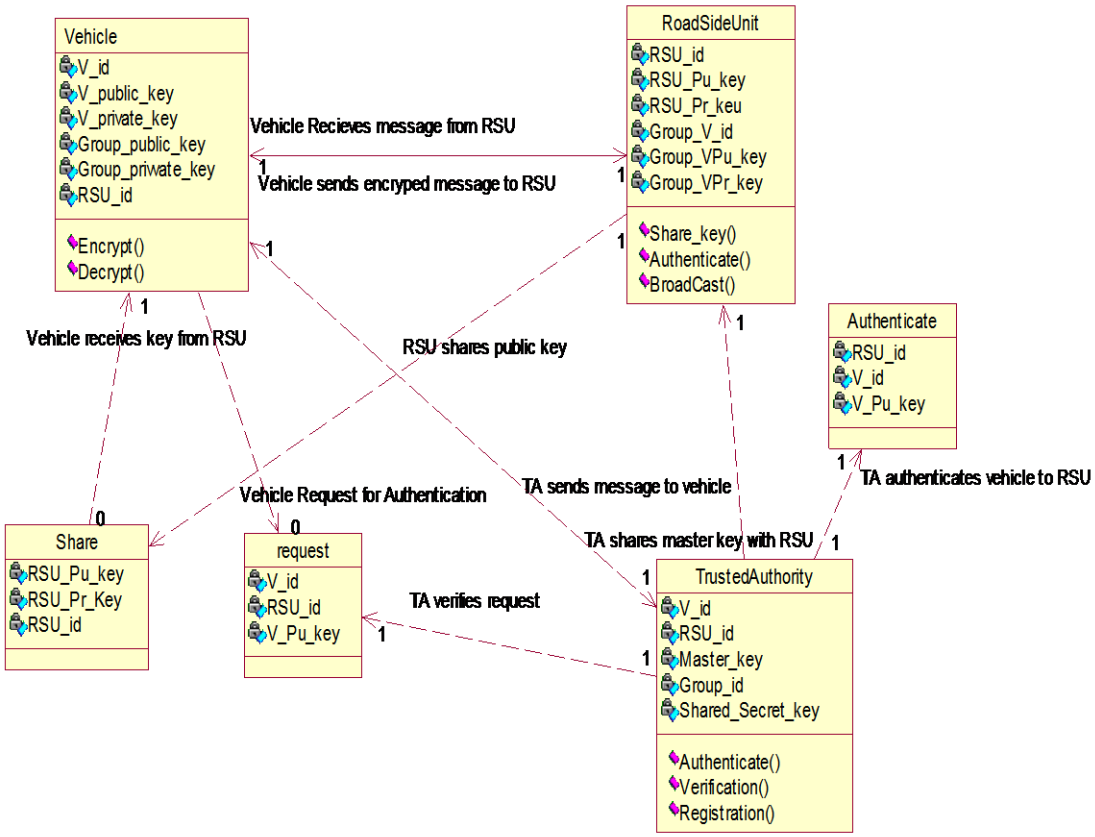

Fig 4.3 Class Diagram for Vehicular Communication in a VANET

Fig.4.3 represents the class diagram of vehicular communications in a VANET. Vehicle first sends its keys to the nearest RSU which in turn sends it to a trusted authority. The trusted authority then authenticates the vehicle public private keys. It then decrypts the received keys of vehicle and stores them in a record. The trusted authority then assigns a group_id to the vehicles within a RSU range. It then shares the public private keys of vehicle and a shared secret key with the RSU. It shares the above keys by encrypting them and sending to RSU. The RSU stores the decrypted keys in a separate record and forward the shared keys it obtained from trusted authority to the vehicle. Thus it completes the registration process of each vehicle in a VANET. The vehicle takes a message which it wants to send to nearest RSU or another vehicle. It then encrypts the message and sends the encrypted message. The

other vehicle then receives the encrypted message and decrypts it to obtain the original message.

Activity Diagram

It is the functional view of a system because it describes logical processes or functions. Each process describes a sequence of tasks and the decisions that govern when and how they are performed.

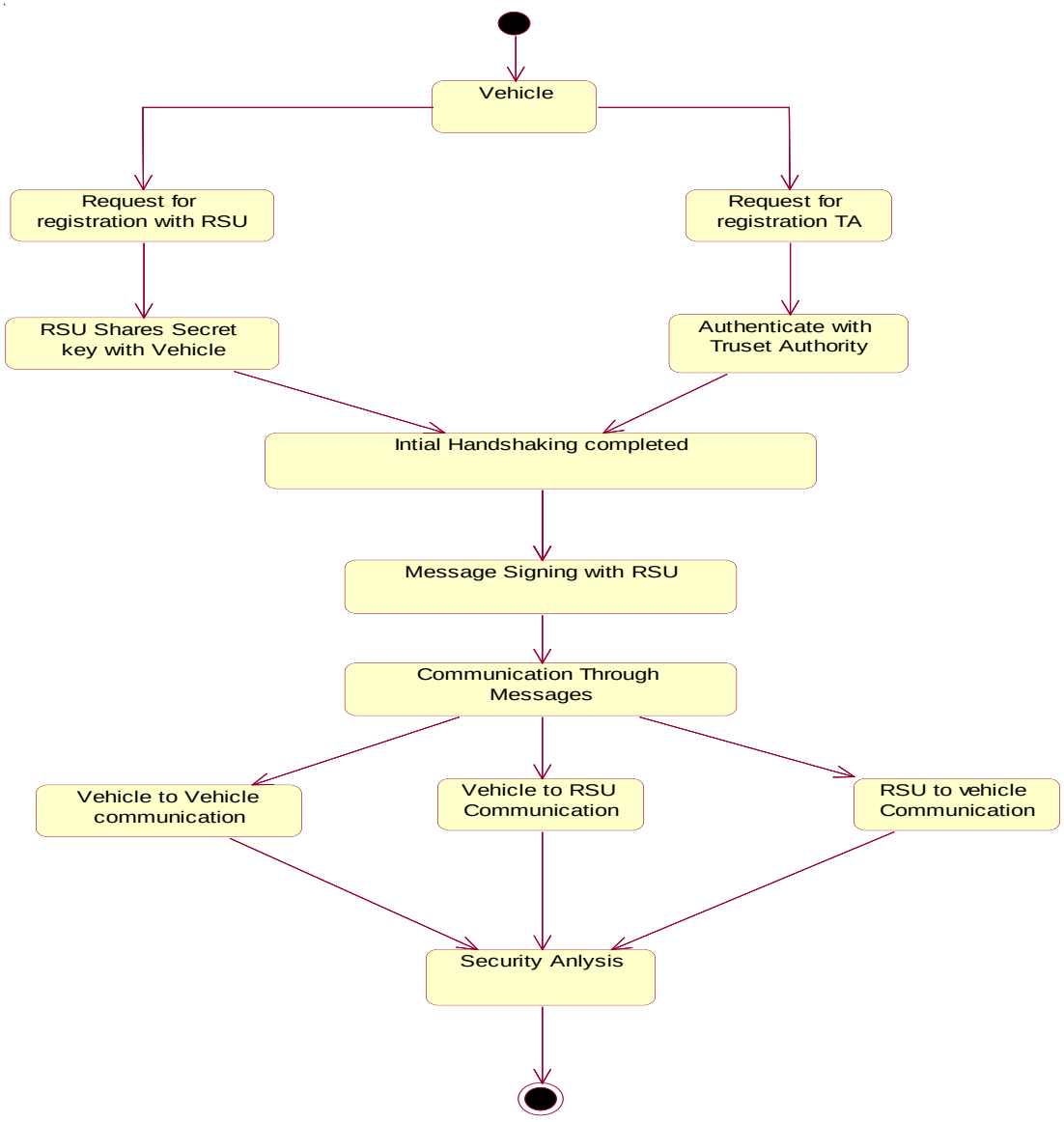

Fig. 4.4 Activity Diagram for After Registration Process

As shown in Fig. 4.4 after the registration process broadcasting of messages takes place. The vehicles can either broadcast messages to other vehicles within their group or with other groups or without a group. After message broadcasting the integrity of the message is checked in security analysis. Thus message broadcasting phase is shown in the above activity diagram.

Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. It is a very well-known adage that "A picture is worth a thousand words". With regards to use case diagrams, that is exactly what they are meant to do.

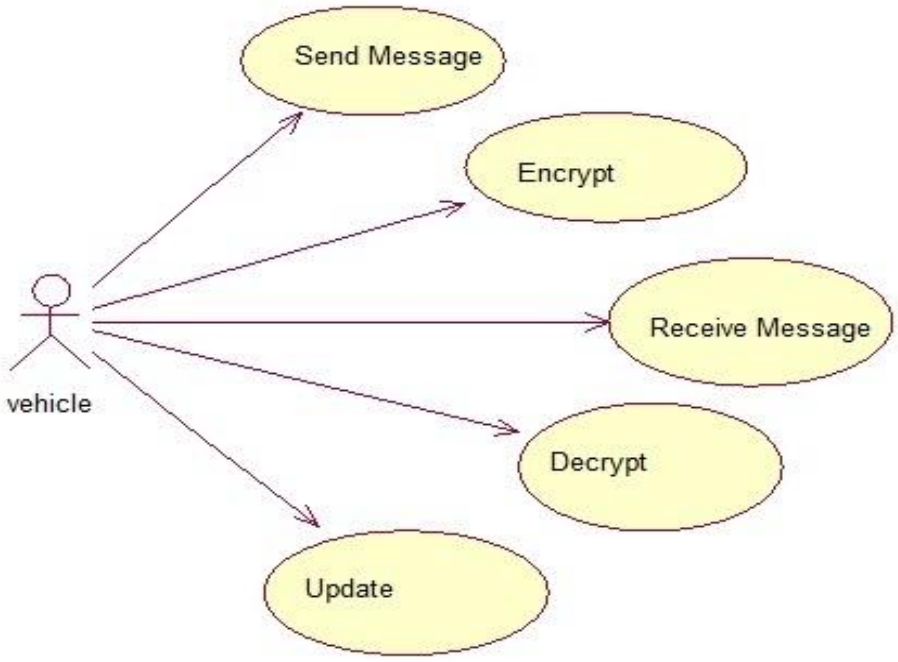

Fig. 4.5 Vehicle Use Case Diagram

Fig. 4.5 represents the use case diagram for a vehicle in a VANET. The vehicle takes a message which it wants to send to nearest RSU or another vehicle. It then encrypts the message and sends the encrypted message. The other vehicle then receives the encrypted message and decrypts it to obtain the original message. It then updates its status and position in the network.

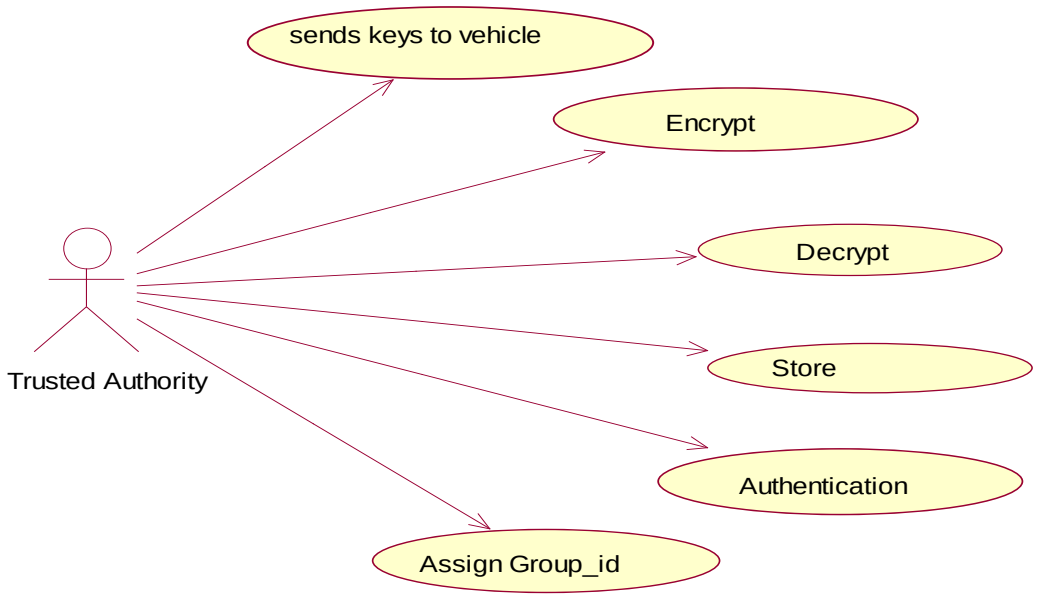

Fig 4.6 Trusted Authority Use Case Diagram

Fig 4.6 shows the use case diagram for trusted authority. Vehicle first sends its keys to the nearest RSU which in turn sends it to a trusted authority. The trusted authority then authenticates the vehicle public private keys. It then decrypts the received keys of vehicle and stores them in a record. The trusted authority then assigns a group_id to the vehicles within a RSU range. It then shares the public private keys of vehicle and a shared secret key with the RSU. It shares the above keys by encrypting them and sending to RSU.

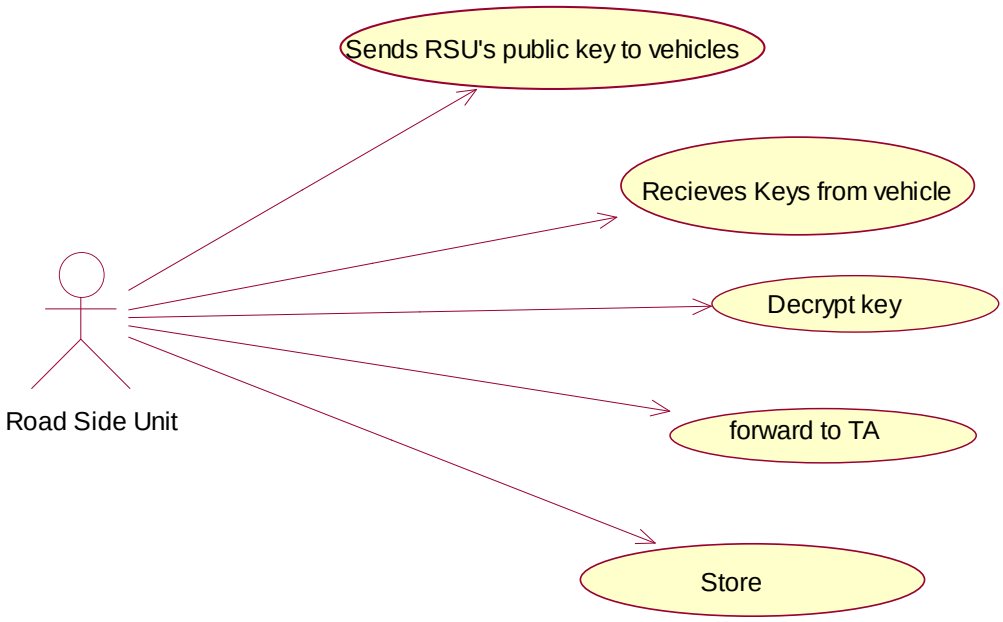

Fig. 4.7 Road Side Unit (RSU) Use Case Diagram

Fig 4.7 shows the use case diagram for Road Side Unit (RSU). The RSU sends it public key to all the vehicles within its range. It then receives public and private keys from vehicles within its range for registration. It decrypts the received keys from vehicle and forwards the keys to a trusted authority. The RSU stores the decrypted keys in a separate record and forward the shared keys it obtained from trusted authority to the vehicle. Thus it completes the registration process of each vehicle in a VANET.

CHAPTER 5

Simulation Study

5.1 Installing NS2

Download ns-allinone-2.35 and Install

$ wget http://nchc.dl.sourceforge.net/sourceforge/nsnam/ns-allinone-2.35.tar.gz

$ tar -xzvf ns-allinone-2.35.tar.gz

$ cd ns-allinone-2.35

$ sudo apt-get install build-essential autoconfautomakelibxmu-dev

edit file /etc/apt/sources.list

$ sudo apt-get install -f build-essential libxt-dev libxt6 libsm-dev libsm6 libice-dev libice6 libxmu-dev

To install NS2:

$./install

Now, the installation has been completed. If you try:

$ ns

Then a "%" will appear on the screen type "exit" to quit the mode and back to "$"

5.2 Modules

The list of all modules present in this implementation process is

1. Creation of Environment and Placing Road Side Unit's
1. Placing Random Nodes in Environment.
2. Generating Public-Private Key Pairs
3. Registration with the RSU
4. Message Broadcasting Step 1
5. Message Broadcasting Step 2

5.2.1 Creation of Environment and Placing Road Side Unit's (RSUs)

RSU is a Road Side Unit. The pseudo code for the creation of environment and RSU is as follows

```
set val(chan)  Channel/WirelessChannel          ;# channel type
set val(prop)   Propagation/TwoRayGround        ;# radio-propagationmodel
set val(netif)  Phy/WirelessPhy                 ;# network interface type
set val(mac)  Mac/802_11                        ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue         ;# interface queue type
set val(ll)       LL                            ;# link layer type
set val(ant)    Antenna/OmniAntenna             ;# antenna model
set val(ifqlen)  20                             ;# max packet in ifq
set numberof     [lindex $argv 0]               ;# number of mobilenodes
set val(rp)     AODV                            ;# routing protocol
set opt(x)     1000                             ;# x coordinate of topology
set opt(y)     1000                             ;# y coordinate of topology
set stopTime    1000.0
LL set mindelay_  50us
LL set delay_    25us
set ns_ [new Simulator]
```

```
set tracefd [open sescpp.tr w]
$ns_ trace-all $tracefd
set namtrace [open sescpp.nam w]
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
set some [gets stdin]
# set up topography object
set topo     [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
# Create God
# create-god $numberof
# Configure node
set chan_1_ [new $val(chan)]
$ns_ node-config
                -mobileIP  OFF\
                -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
                 -channel $chan_1_


for {set i 0} {$i < $numberof} {incr i} {
```

```
  set node_($i) [$ns_ node]
    if {$i<12} {
         $node_($i) shape "square"
    }
  $node_($i) random-motion 0  ;# disable random motion
}
```

Creation of Road Side Units (RSU):

```
$node_(0) set X_ 200              # Define x co-ordinate
$node_(0) set Y_ 900              # Define y co-ordinate
$node_(0) set Z_ 0.0              # Define z co-ordinate
$node_(1) set X_ 550
$node_(1) set Y_ 900
$node_(1) set Z_ 0.0
$node_(2) set X_ 900
$node_(2) set Y_ 900
$node_(2) set Z_ 0.0
$node_(3) set X_ 900
$node_(3) set Y_ 670
$node_(3) set Z_ 0.0
$node_(4) set X_ 900
$node_(4) set Y_ 100
$node_(4) set Z_ 0.0
$node_(5) set X_ 550
$node_(5) set Y_ 100
$node_(5) set Z_ 0.0
$node_(6) set X_ 200
$node_(6) set Y_ 100
$node_(6) set Z_ 0.0
$node_(7) set X_ 350
$node_(7) set Y_ 200
```

$node_(7) set Z_ 0.0

$node_(8) set X_ 700

$node_(8) set Y_ 200

$node_(8) set Z_ 0.0

$node_(9) set X_ 800

$node_(9) set Y_ 430

$node_(9) set Z_ 0.0

$node_(10) set X_ 700

$node_(10) set Y_ 800

$node_(10) set Z_ 0.0

$node_(11) set X_ 400

$node_(11) set Y_ 800

$node_(11) set Z_ 0.0

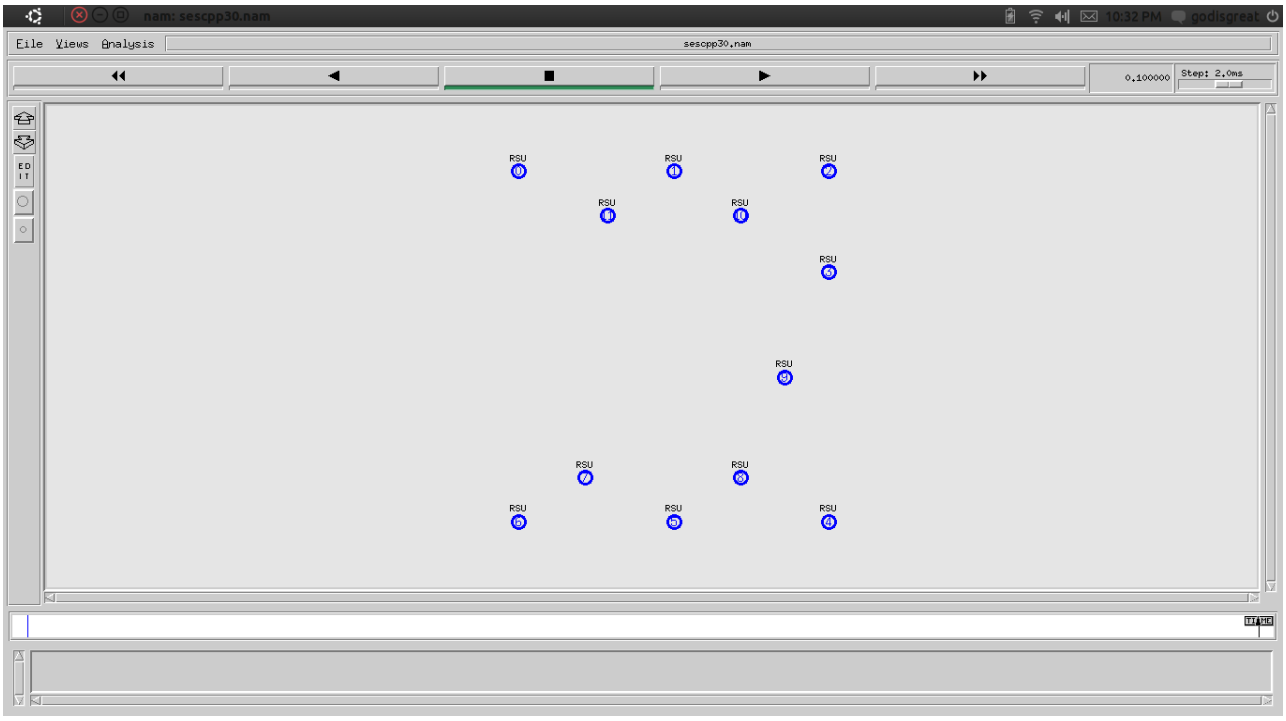

Fig. 5.1 Creation of Environment with 12 RSUs

Fig. 5.1 shows the creation of Environment with 12 Road Side Units which are static and are capable of forming groups with their own with their own public and private keys.

5.2.2 Creation of Random Nodes:

```
for {set i 12} {$i<$upMax} {incr i} {
        $node_($i) set X_ 0.0
        $node_($i) set Y_ 880
        $node_($i) set Z_ 0.0
        $node_($i) color green
        set vNo [expr ($i-12)]
        $ns_ at 0.00 "$node_($i) label V$vNo"
        $ns_ at 0.0 "$node_($i) color  green"
        $ns_ initial_node_pos $node_($i) 10
        #$node_($i) setdest 0.0 120 1 }
for {set i $upMax} {$i<$numberof} {incr i} {
        $node_($i) set X_ 0.0
        $node_($i) set Y_ 180
        $node_($i) set Z_ 0.0
        $node_($i) color  pink
        set vNo [expr ($i-12)]
        $ns_ at 0.05 "$node_($i) label V$vNo"
        $ns_ at 0.05 "$node_($i) color  pink"
        $ns_ initial_node_pos $node_($i) 10
        #$node_($i) setdest 0.0 820 1
}
```

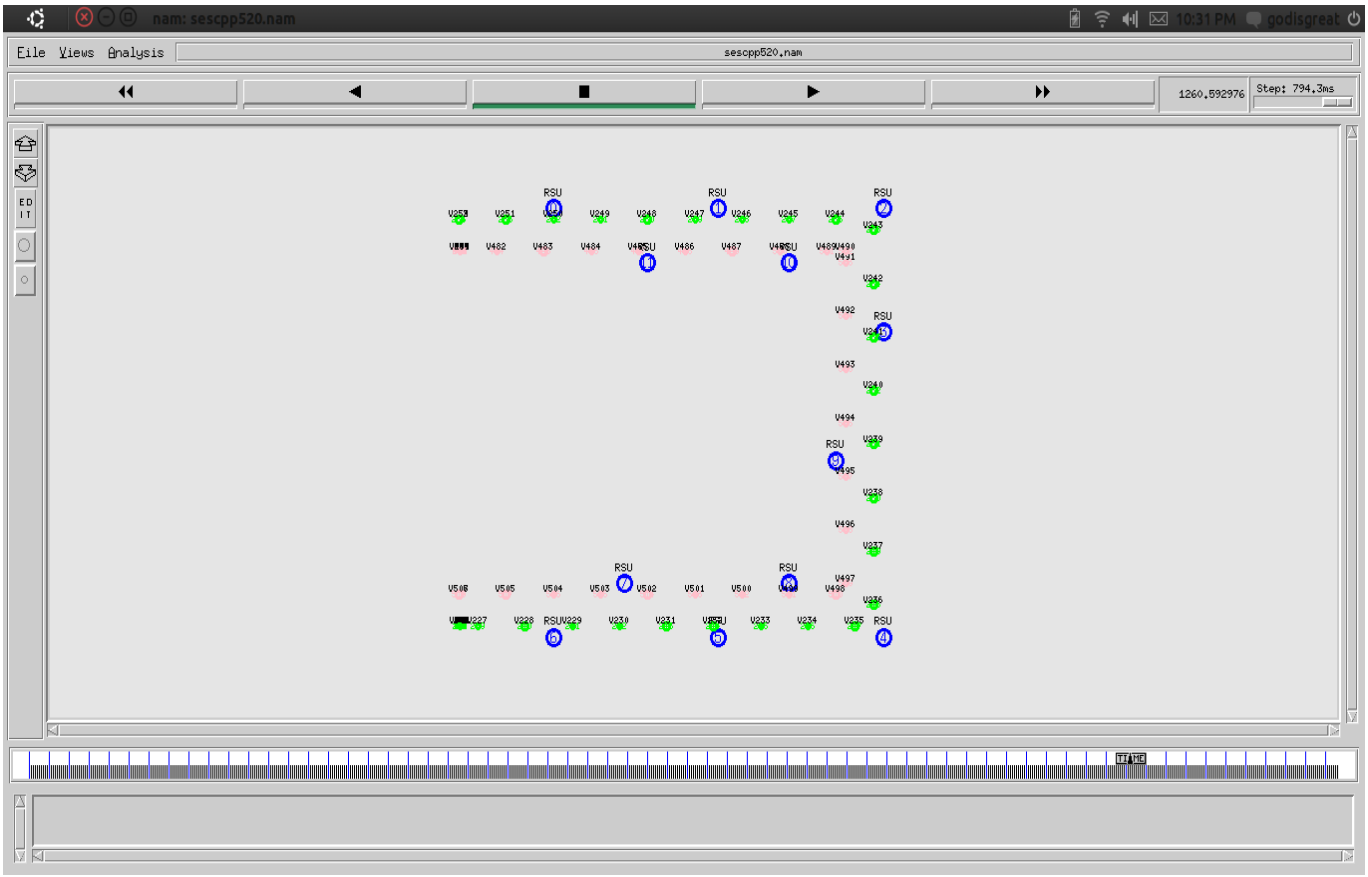

Fig. 5.2 Creation of 500 Random Nodes

Fig. 5.2 shows the creation of topology with 500 nodes having the random moment, randomly assigned co-ordinates.

5.2.3 Generating Public-Private Key Pairs

Each vehicle or node will be given its own public-private keys in addition to the group public-private keys. Both are RSA key pairs. The key pair that is obtained through access point (RSU) is called group key. The one that is unique for each node is called identity key pair.

Public Key Generating Procedure

```
procPublickeyGen{name1 name2 name3} {   #Procedure initialization

set n [expr $name1*$name2]                    #Multiplying the parameters
```

```
set pi [expr ($name1-1)*($name2-1)]            #Euler's function
set e $name3
for {set i 1} {$i < 10000 } {incri} {
if {[ expr ($i*$e)%$pi] == 1 } {                #Generating values
set d $i
return $d                                       #Procedure completed
} } }
```

Private Key Generating Procedure

```
procPrivatekeyGen{key1 key2 key3} {            #Procedure initialization
set n [expr $key1*$key2]                       #Multiplying the parameters
set pi [expr ($key1-1)*($nkey2-1)]             #Euler's function
set e $key3                                    #Generating Public key
for {set i 1} {$i < 10000 } {incri} {
if {[ expr ($i*$e)%$pi] == 1 } {
set d $i
return $d                                      #Procedure completed
} } }
```

Above procedures will generate the public-private key pairs.

5.2.4 Registration with the RSU

The vehicle or node has to register with the RSU to enter the region of that RSU. The vehicle authenticated itself with the Trusted authority via RSU

Initially the vehicle starts the registration procedure. The vehicle sends its public key and its signature to the closest RSU. RSU verify the vehicle's signature even if it uses the pseudo identity to sign the message. RSU generates a shared secret key with the vehicle in the first time vehicle comes into that region for security purpose shared secret key not preloaded into any hardware on vehicle. A new secret key is generated every time the vehicle moves into the new region. The node has to register with the RSU to obtain the group public-private key pair for group communication.

Bloom Filter Technique:

The code for the Bloom Filter Technique is as follows

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cstring>
using namespace std;
#define POLYNOMIAL 0x04C11DB7L // Standard CRC-32 polynomial
#define M 32   // Number of bits in the Bloom filter
#define K 4   // Number of bits set per mapping in filter
typedef unsigned short int word16;
typedef unsigned int word32;
static word32 CrcTable[256];       // Table of 8-bit CRC32 remainders
char BFilter[M / 8];                  // Bloom filter array of M/8 bytes
word32 NumBytes;             // Number of bytes in Bloom filter
void gen_crc_table(void);
word32 update_crc(word32 crc_accum, char *data_ptr, word32 data_size);
void mapBloom(word32 hash);
int testBloom(word32 hash);
```

```
int main()
{
FILE *fp1;
FILE *fp2;
char inFile1[256];
char inFile2[256];
char inString[1024];
word32 crc32;
int retCode;
word32 i;
// Determine number of bytes in Bloom Filter
NumBytes = M / 8;
if ((M % 8) != 0)
{
exit(1);
}
// Clear the Bloom filter
for (i = 0; i < NumBytes; i++)
BFilter[i] = 0x00;
// Initialize the CRC32 table
gen_crc_table();
cin>>inFile1;                          #File name of input list to add to filter
fp1 = fopen(inFile1, "r");
if (fp1 == NULL)
{
exit(1);
}
```

```
cin>>inFile2;                          #File name of input list to check for match
fp2 = fopen(inFile2, "r");
if (fp2 == NULL)
{
exit(1);                               #ERROR in opening input file #1 ("<<inFile2<<")
 }
// Read input file #1 and map each string to Bloom filter
while (1)
{
fgets(inString, 1024, fp1);
if (feof(fp1))
break;
for (i = 0; i < K; i++)
{
crc32 = update_crc(i, inString, strlen(inString));
mapBloom(crc32);
}
}
fclose(fp1);
for (i = 0; i < NumBytes; i++)
printf("%2d", i);
cout<<endl;
for (i = 0; i < NumBytes; i++)
printf("%02X", BFilter[i]);
cout<<endl;
while(1)
{
```

```
fgets(inString, 1024, fp2);
if (feof(fp2))
break;
for (i = 0; i < K; i++)
{
crc32 = update_crc(i, inString, strlen(inString));
retCode = testBloom(crc32);
                    if (retCode == 0)
                            break;
.           }
            if (retCode == 1)
                cout<<inString;
          }
          fclose(fp2);
        }
void gen_crc_table(void)
{
register word32 crc_accum;
register word16 i, j;
// Initialize the CRC32 8-bit look-up table
for (i = 0; i < 256; i++)
{
crc_accum = ((word32) i << 24);
for (j = 0; j < 8; j++)
{
if (crc_accum & 0x80000000L)
crc_accum = (crc_accum << 1) ^ POLYNOMIAL;
```

```
else
crc_accum = (crc_accum << 1);
}
CrcTable[i] = crc_accum;
}
}
word32 update_crc(word32 crc_accum, char *data_blk_ptr, word32 data_blk_size)
{
register word32 i, j;
for (j = 0; j < data_blk_size; j++)
{
i = ((int) (crc_accum >> 24) ^ *data_blk_ptr++) & 0xFF;
crc_accum = (crc_accum << 8) ^ CrcTable[i];
}
crc_accum = ~crc_accum;
return crc_accum;
}
// Function to map hash into Bloom filter
void mapBloom(word32 hash)
{
int tempInt;
int bitNum;
int byteNum;
unsigned char mapBit;
tempInt = hash % M;
byteNum = tempInt / 8;
```

```
bitNum = tempInt % 8;
mapBit = 0x80;
mapBit = mapBit >> bitNum;
BFilter[byteNum] = BFilter[byteNum] | mapBit;
}
int testBloom(word32 hash)              #Function to test for a Bloom filter match
{
int tempInt;
int bitNum;
int byteNum;
unsigned char testBit;
int retCode;
tempInt = hash % M;
byteNum = tempInt / 8;
bitNum = tempInt % 8;
testBit = 0x80;
testBit = testBit >> bitNum;
if (BFilter[byteNum] & testBit)
retCode = 1;
else
retCode = 0;
return retCode;
}
```

The code for the SHA-1 is as follows

```
#include “sha1.h”
```

```
#include<sstream>
#include<iomanip>
#include<fstream>
SHA1::SHA1()
{
reset();
}
void SHA1::update(const std::string &s)
{
std::istringstream is(s);
update(is);
}
void SHA1::update(std::istream &is)
{
std::string rest_of_buffer;
read(is, rest_of_buffer, BLOCK_BYTES - buffer.size());
buffer += rest_of_buffer;
while (is)
{
uint32 block[BLOCK_INTS];
buffer_to_block(buffer, block);
transform(block);
read(is, buffer, BLOCK_BYTES);
}
}


// Add padding and return the message digest.
std::string SHA1::final()
{
```

```
//Total number of hashed bits
uint64 total_bits = (transforms*BLOCK_BYTES + buffer.size()) * 8;
buffer += 0x80;                         //Padding
unsigned int orig_size = buffer.size();
while (buffer.size() < BLOCK_BYTES)
{
buffer += (char)0x00;
}
uint32 block[BLOCK_INTS];
buffer_to_block(buffer, block);
if (orig_size > BLOCK_BYTES - 8)
{
transform(block);
for (unsigned int i = 0; i < BLOCK_INTS - 2; i++)
{
block[i] = 0;
}
}
// Append total_bits, split this uint64 into two uint32
block[BLOCK_INTS - 1] = total_bits;
block[BLOCK_INTS - 2] = (total_bits >> 32);
transform(block);
std::ostringstream result;
for (unsigned int i = 0; i < DIGEST_INTS; i++)
{
result << std::hex << std::setfill('0') << std::setw(8);
result << (digest[i] & 0xffffffff);
}
// Reset for next run
reset();
return result.str();
```

```
}
std::string SHA1::from_file(const std::string &filename)
{
std::ifstream stream(filename.c_str(), std::ios::binary);
SHA1 checksum;
checksum.update(stream);
return checksum.final();
}
void SHA1::reset()
{
// SHA1 initialization constants
digest[0] = 0x67452301;
digest[1] = 0xefcdab89;
digest[2] = 0x98badcfe;
digest[3] = 0x10325476;
digest[4] = 0xc3d2e1f0;
// Reset counters
transforms = 0;
buffer = "";
}
//Hash a single 512-bit block. This is the core of the algorithm.
void SHA1::transform(uint32 block[BLOCK_BYTES])
{
/* Copy digest[] to working vars */
uint32 a = digest[0];
uint32 b = digest[1];
uint32 c = digest[2];
uint32 d = digest[3];
uint32 e = digest[4];
// Help macros
#define rol(value, bits) (((value) << (bits)) | (((value) & 0xffffffff) >> (32 - (bits))))
```

```
#define blk(i) (block[i&15] = rol(block[(i+13)&15] ^ block[(i+8)&15] ^ block[(i+2)&15] ^ block[i&15],1))
// (R0+R1), R2, R3, R4 are the different operations used in SHA1
#define R0(v,w,x,y,z,i)   z+=((w&(x^y))^y)+block[i]+0x5a827999+rol(v,5);
w=rol(w,30);
#define R1(v,w,x,y,z,i)   z+=((w&(x^y))^y)+blk(i)+0x5a827999+rol(v,5);
w=rol(w,30);
#define R2(v,w,x,y,z,i) z += (w^x^y) +blk(i)+0x6ed9eba1+rol(v,5);
 w=rol(w,30);
#defineR3(v,w,x,y,z,i)z+=(((w|x)&y)|(w&x))+blk(i)+0x8f1bbcdc+rol(v,5);
w=rol(w,30);
#define R4(v,w,x,y,z,i) z += (w^x^y)+blk(i)   + 0xca62c1d6 + rol(v,5); w=rol(w,30);
// 4 rounds of 5 operations each. Loop unrolled.
R0(a,b,c,d,e, 0);
R0(e,a,b,c,d, 1);
R0(d,e,a,b,c, 2);
R0(c,d,e,a,b, 3);
R0(a,b,c,d,e, 4);
R1(e,a,b,c,d,5);
R1(d,e,a,b,c,6);
R1(c,d,e,a,b,7);
R1(b,c,d,e,a,8);
R2(a,b,c,d,e,9);
R2(e,a,b,c,d,10);
R2(d,e,a,b,c,11);
R2(c,d,e,a,b,12);
R2(b,c,d,e,a,13);
R2(a,b,c,d,e,14);
R3(a,b,c,d,e,15);
R3(e,a,b,c,d,16);
R3(d,e,a,b,c,17);
```

```
R3(c,d,e,a,b,18);
R3(b,c,d,e,a,19);
R4(a,b,c,d,e,20);
R4(e,a,b,c,d,21);
R4(d,e,a,b,c,22);
R4(c,d,e,a,b,23);
R4(b,c,d,e,a,24);
// Add the working vars back into digest[]
digest[0] += a;
digest[1] += b;
digest[2] += c;
digest[3] += d;
digest[4] += e;
// Count the number of transformations
transforms++;
}
VoidSHA1::buffer_to_block(conststd::string&buffer,uint32block[BLOCK_BYTES)
{
// Convert the std::string (byte buffer) to a uint32 array (MSB)
for (unsigned int i = 0; i < BLOCK_INTS; i++)
{
block[i] = (buffer[4*i+3] & 0xff)
| (buffer[4*i+2] & 0xff)<<8
| (buffer[4*i+1] & 0xff)<<16
| (buffer[4*i+0] & 0xff)<<24;
}
}
void SHA1::read(std::istream &is, std::string &s, int max)
{
char sbuf[max];
is.read(sbuf, max);
```

```
s.assign(sbuf, is.gcount());
}
```

5.2.5 Message Broadcasting Step 1

Any vehicle can broadcast a message to nearest RSU. It first creates a pseudo identity together with the signing key it signs the message using of the pseudo identity. Upon receiving the message, the RSU finds out vehicles public key and shared secret key. RSU to verify a batch of signatures using only two pairing operations based on the bilinear property and bilinear map and also RSU perform batch verification for sending of safety messages to other. To reduce the message over head we use of bloom filter. We use the UDP or TCP protocol to broadcast the message.

The pseudo code for the broadcasting of messages between vehicle and an Road side units (RSU)

```
Agent/TCP set window_ 20
Agent/TCP set packetSize_ 100
Agent/TCP set maxburst_ 0
Agent/TCP set maxcwnd_ 0
Agent/TCP set ts_option_size_ 20;     # in bytes
Agent/UDP set packetSize_ 100
Application/Traffic/CBR set rate_ 64Kb
Application/Traffic/CBR set random_ NO
Application/Traffic/CBR set interval_ 16
Application/Traffic/CBR set maxpkts_ 2280000
set tcp0 [new Agent/TCP]
$tcp0 set class_ 2
set sink0 [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $tcp0
$ns_ attach-agent $node_(12) $sink0
$ns_ connect  $tcp0 $sink0
set ftp0 [new Application/FTP]
```

$ftp0 attach-agent $tcp0

$ns_ at 2.0 "$ftp0 start"

$ns_ at 300.0 "$ftp0 stop"

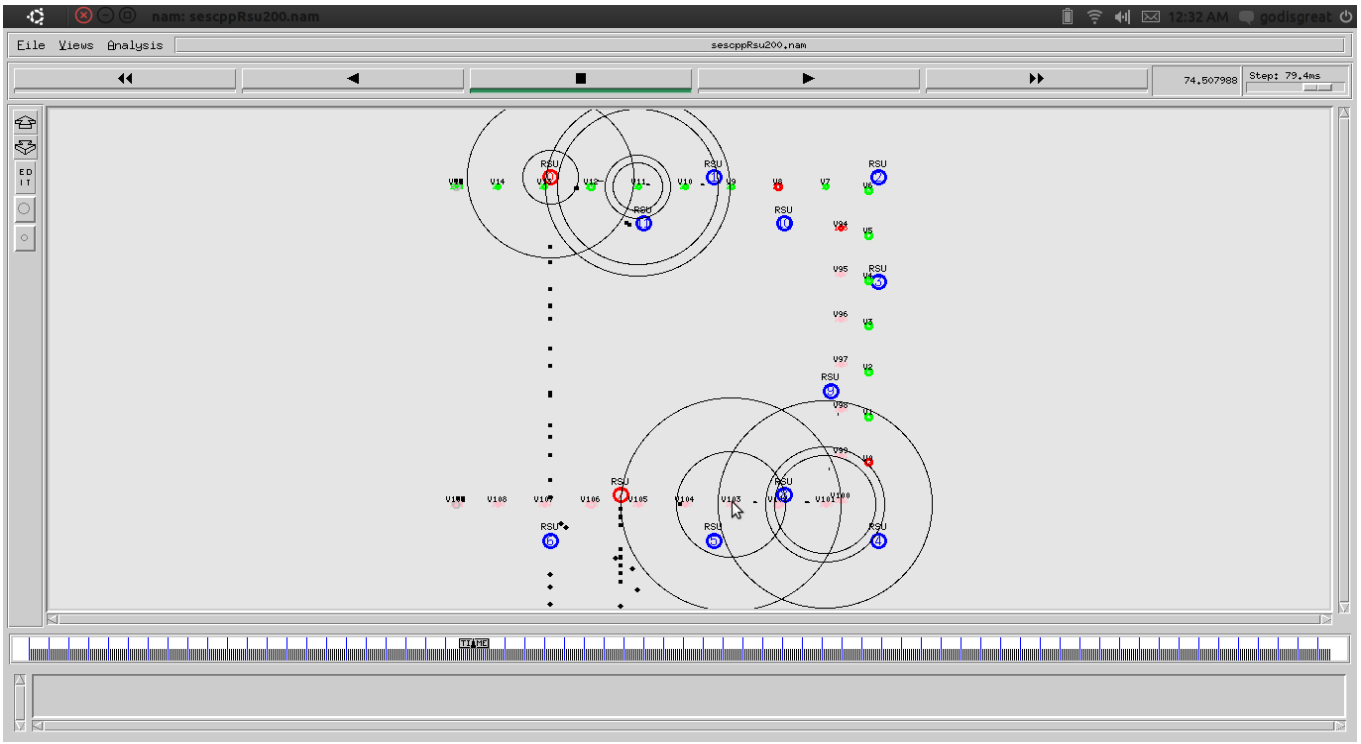

Fig. 5.3 Vehicle to Road Side Unit Communication

Fig.5.3 Shows the Communication between Vehicles to RSU. In the above figure vehicles 9, 10, 11, 12 are sending messages to RSU1 and vehicles 104, 105, 106, 107 are sending messages to RSU3. Data packets are represented in dotted line as shown above. Circles show the range of vehicle up to which it can transmit message to new RSU.

5.2.6 Message Broadcasting Step 2

We assumed that vehicles have already formed a group and have their group public keys. Here in one to one communications within a group occurs as a vehicle sends a message to another vehicle. The vehicle sends a message to another vehicle of that without involvement of an RSU of the same group. Same group of vehicles have the group public key and also group private key.

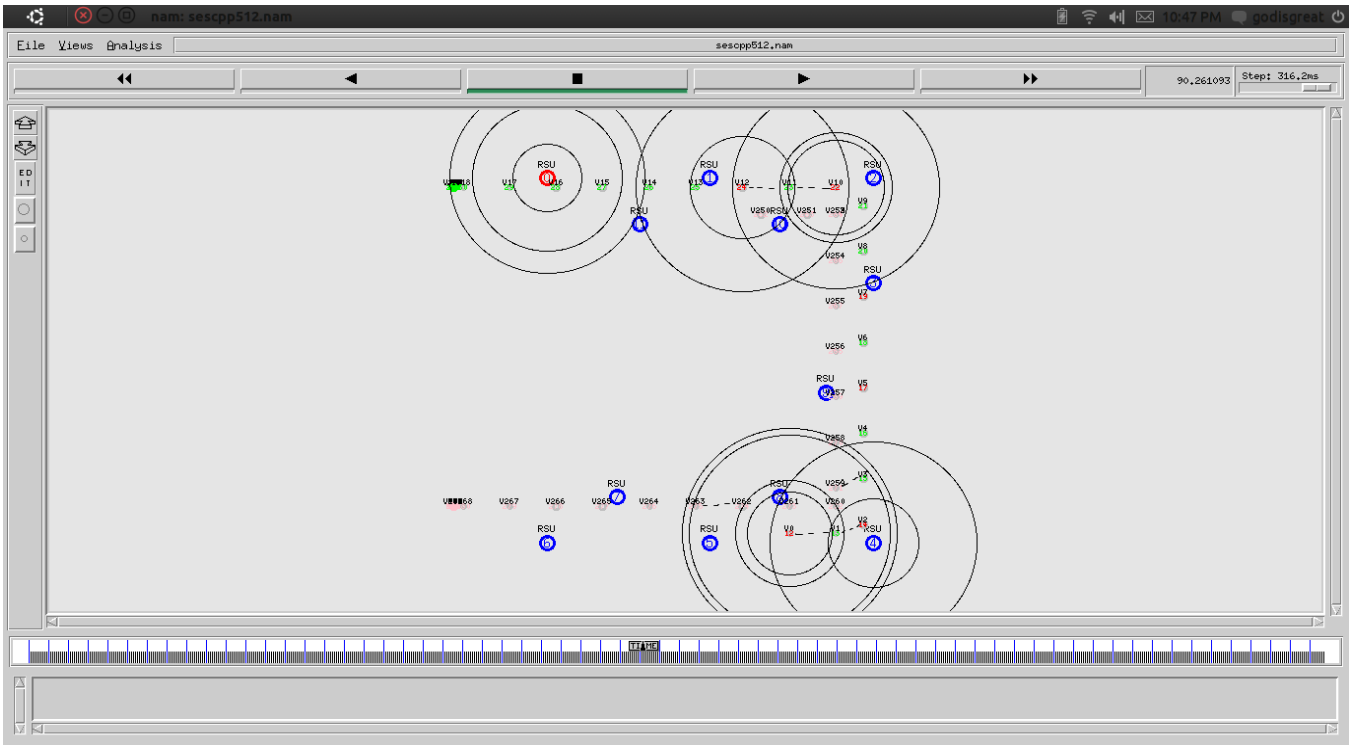

Fig. 5.4 Vehicle to Vehicle Communication

Fig.5.4 Shows the Communication between Vehicles to Vehicle. The above figure shows vehicle 11 sends messages to vehicle 10, vehicle 12, vehicle 259 sends message to vehicle 3, vehicle 263 sends message to vehicle 262, and Data packets are represented in dotted line as shown above. Circles show the range of vehicle up to which it can transmit message to another vehicle.

CHAPTER 6

Experimental Results

6.1 Theoretical Analysis of Simulation Study

The scheme has implemented by using NS2 (Network Simulator 2.34 version). All the integrated techniques of our scheme have implemented by using C++. Assume that an RSU is installed on a highway and vehicles pass through it at speeds varying from 50 km/h to 70 km/h. The RVC and the IVC ranges are set to 600 m and 300 m respectively. That is, when a vehicle enters the 600 m RVC range of the RSU, the messages sent by it can received by the RSU and at the same time, the messages sent by the RSU can be received by it. Inter-vehicle messages are sent every 500 ms at each vehicle.

IEEE 802.11a is used to simulate the medium access control layer. That is, when a vehicle wants to transmit, it first detects whether the channel is available. If another vehicle is transmitting, it waits until that transmission is completed and then waits for a random delay period before it begins to transmit. The bandwidth of the channel is 6 Mb/s and the average length of inter-vehicle message is 200 bytes. Each pairing operation takes 4.5ms. The simulation runs for 1000 s. First vary the total number of vehicles that have ever entered RSU's RVC range during the simulation period from 200 to 1000 in steps of 200 to simulate the impact of different traffic densities. We then vary the inter-vehicle message signature error rate from 1% to 10% to interpret its impact on the performance of our schemes. Finally we vary the RSU's batch verification period from 100 ms to 1000 ms in steps of 100 ms to investigate its impact on different schemes. For each configuration, compute the average of five different random scenarios.

6.2 Experiments

6.2.1 Experiment 1:

Table 6.1 Delay(s) values for IBV protocol and Different levels of Binary Searching Techniques

Number of Vehicles	IBV Protocol	Binary Search(0)	Binary Search(1)	Binary Search(2)	Binary Search(3)	Binary Search(4)
100	0.0120	0.0090	0.0090	0.0102	0.0115	0.0130
200	0.0122	0.0092	0.0095	0.0105	0.0120	0.013.8
300	0.0125	0.0093	0.0100	0.0108	0.0133	0.0151
400	0.0128	0.0095	0.0105	0.0112	0.0145	0.0155
500	0.0129	0.0096	0.0107	0.0115	0.0150	0.0165
600	0.0130	0.0098	0.0110	0.0120	0.0155	0.0170
700	0.0132	0.0100	0.0113	0.0126	0.0160	0.0175
800	0.0133	0.0102	0.0115	0.0129	0.0165	0.0182
900	0.0135	0.0104	0.0118	0.0135	0.0170	0.0185
1000	0.014	0.0105	0.0120	0.0145	0.0175	0.0190

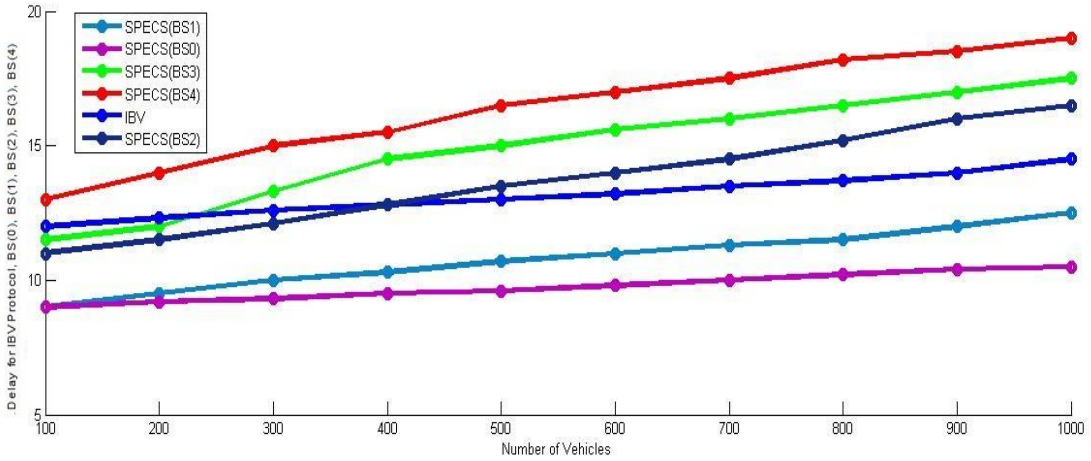

Fig. 6.1 Delay for IBV Protocol and SPECS (BS(0), BS(1), BS(2), BS(3), BS(4)).

After simulation the obtained values of delay for IBV Protocol and Different levels of Binary searching Technique ((BS(0), BS(1), BS(2), BS(3), BS(4)) are plotted in the above graph by using MATLAB. From the above graph, Delay under the IBV protocol and provided scheme are close to each other. The delay values are increasing for higher levels of binary search. The Delay value for single level of pairing is 2ms less than IBV Protocol.

6.2.2 Experiment 2:

Table 6.2 Success Rate for IBV protocol and Different levels of Binary Searching Techniques

Number of Vehicles	IBV Protocol	Binary Search(1)	Binary Search(2)	Binary Search(3)	Binary Search(4)
100	0	65	90	95	97
200	0	63	88	92	96
300	0	60	85	90	95
400	0	58	82	87	93
500	0	55	80	84	91
600	0	53	78	83	90
700	0	21	76	82	88
800	0	20	75	81	87
900	0	44	73	80	86
1000	0	40	70	78	85

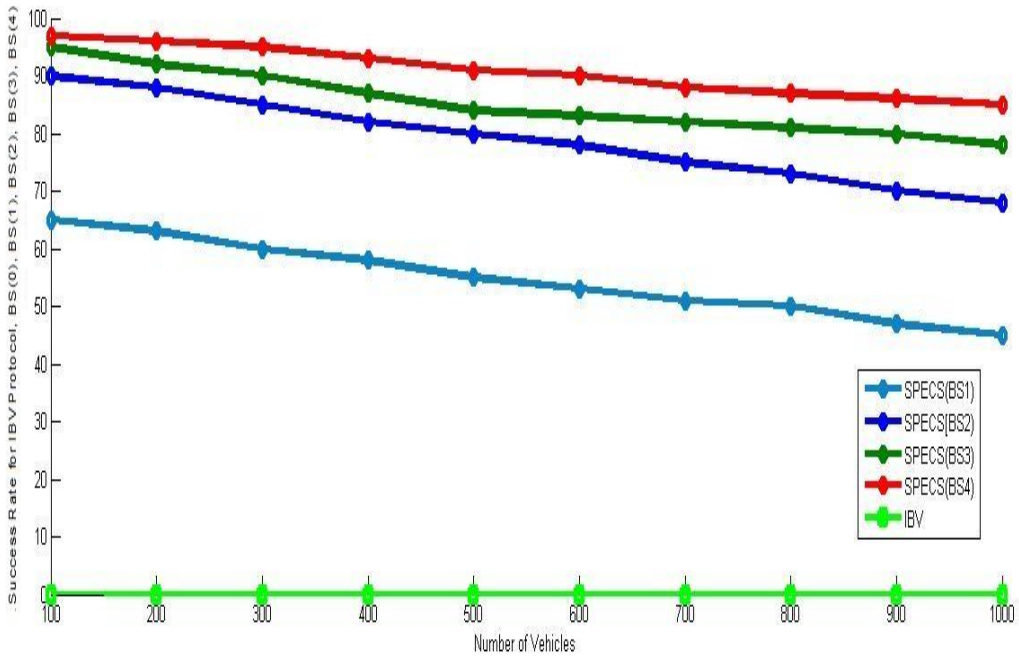

Fig. 6.2 Success Rate for IBV Protocol and SPECS(BS(0), BS(1), BS(2), BS(3), BS(4)).

After simulation the obtained values of Success Rate for IBV Protocol and Different levels of Binary searching Technique ((BS(0), BS(1), BS(2), BS(3), BS(4)) are plotted in the above graph by using MATLAB. From the above graph, Success Rate for the proposed Scheme is 40% higher Successful rate than IBV Protocol. The Success Rate for the IBV protocol is 0 because in batch verification IBV protocol drops the whole batch of signatures when a single signature is invalid.

6.2.3 Experiment 3:

Table 6.3 Delay, Error Rate and Success Rate values for Vehicle to Vehicle communication

Number of Vehicles	Delay(s)	Error Rate (%)	Success Rate (%)
100	0.0130	7.00	95
200	0.0135	7.20	93.00
300	0.0138	7.40	92.00
400	0.0140	7.80	91.00
500	0.0145	8.30	89.50
600	0.0150	8.50	87.00
700	0.0153	8.70	86.50
800	0.0158	9.00	84.05
900	0.0160	9.20	83.00
1000	0.0170	9.30	80.50

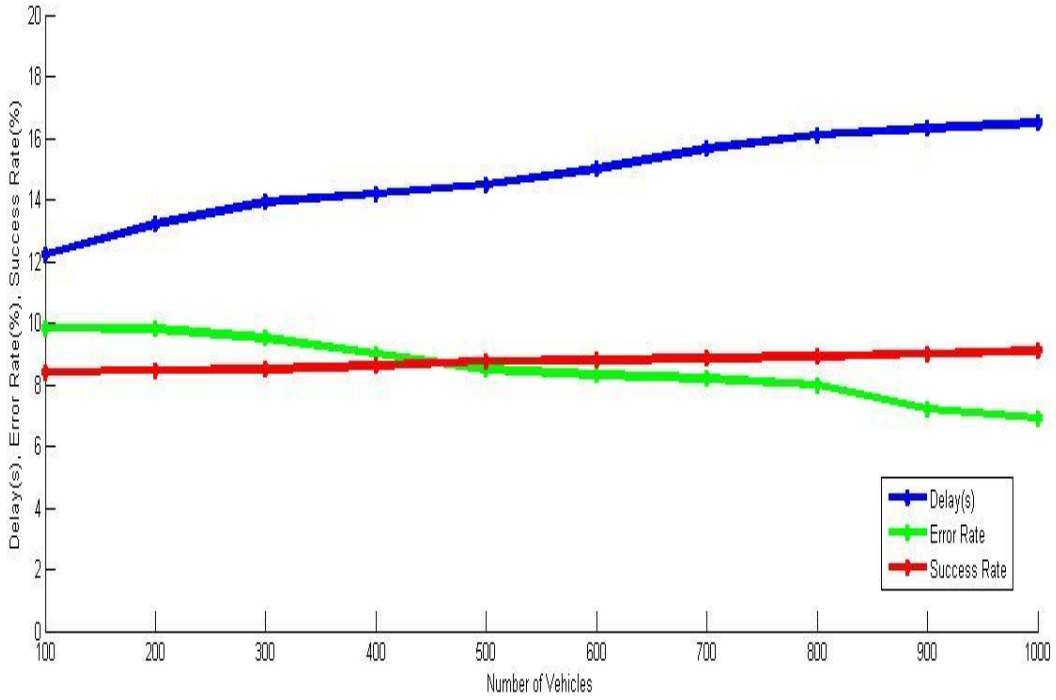

Fig. 6.3 Delay, Error Rate and Success Rate for Vehicle to Vehicle Communication

After simulation the obtained values of Delay, Error Rate and Success Rate for Vehicle to Vehicle communication are plotted in the above graph by using MATLAB. From the above graph, Delay and Error Rate values are increasing when number of vehicles increasing. Success Rate values are decreasing when Error Rate increasing.

6.2.4 Experiment 4:

Table 6.4 Delay, Error Rate, Success Rate values for communication between Vehicle to RSU

Number of Vehicles	Delay(s)	Error Rate (%)	Success Rate (%)
100	0.0140	6.80	97
200	0.0145	7.00	96.00
300	0.0155	7.80	95.00
400	0.0160	7.80	93.00
500	0.0170	8.30	91.50
600	0.0175	8.60	90.00

700	0.0180	9.00	88.50
800	0.0183	9.30	87.05
900	0.0185	9.50	86.00
1000	0.0190	9.80	85.50

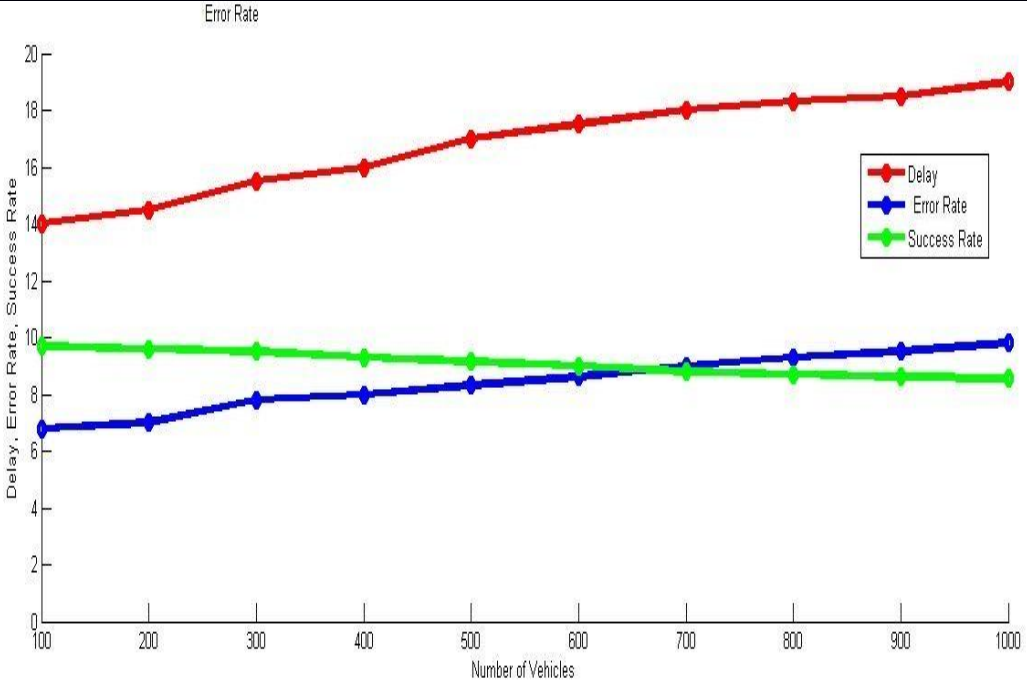

Fig. 6.4 Delay, Error Rate and Success Rate for Vehicle to RSU Communication

After simulation the obtained values of Delay, Error Rate and Success Rate for Vehicle to Vehicle communication are plotted in the above graph by using MATLAB. From the above graph, Delay and Error Rate values are increasing when number of vehicles increasing. Success Rate values are decreasing when Error Rate increasing.

6.3 Simulation Results

The obtained values of delay, error rate and success rate are plotted in the above graphs. The results show that the Delay is acceptable and higher successful rate than Existing solutions.

CHAPTER 7

Conclusion and Future Work

7.1 Work Done

The project provides a more comprehensive set of methods for VANETs. This method satisfies all security requirements for VANETs. This project provides two secure and privacy enhancing communications schemes for VANETs to handle ad hoc messages and group messages for inter-vehicle communications. In terms of effectiveness, this project gives a solution of lower message overhead. In this project, bloom filter and binary search techniques are used for RSU message verification and reduction of message overhead. This project first provides a group communication protocol to allow vehicles to form a group and communicate securely. The scheme is used to communicate vehicles in the same group without RSU. This scheme is used for vehicles to send and receive messages more securely and confidentially.

7.2 Future Work

There is a large scope for advancement in the field of vehicular ad hoc networks. Due to its highly dynamic nature VANET's environment presents great challenges in designing appropriate protocols. There is a scope for future work in the following

- Extend group communication protocol to allow dynamic membership.
- Make strategies to avoid Sybil attacks at a low cost.

Chapter 8

References

[1] T.W. Chim, S.M. Yiu, Lucas C.K Hui, Victor O.K Li, "SPECS: Secure and Privacy Enhancing Communications Scheme for VANETs", *Ad Hoc Networks,* pp. 189-203, September 2011.

[2] H. Oh, C. Yae, D. Ahn, H. Cho, "DSRC Packet Communication System for ITS Services", *Proceedings of the IEEE VTC,* pp. 2223-2227, September 1999.

[3] D. Boneh, B. Lynn, H. Shacham, "Short Signatures from the Well Pairing", *Proceedings of Asiacrypt*, pp. 514-532, January 2001.

[4] P.P. Tsang, S.W. Smith, "PPAA: Peer-to-Peer Anonymous Authentication", *Proceedings of ACNS*, pp. 55–74, August 2008.

[5] C. Zhang, X. Lin, R. Lu, P.H. Ho, "RAISE: An Efficient RSU-Aided Message Authentication Scheme in Vehicular Communication Networks", *Proceedings of the IEEE ICC*, pp. 1451–1457, May 2008.

[6] D. Pointcheval, J. Stern, "Security Arguments for Digital Signature and

Blind Signatures", *journal of Cryptography,* pp. 123-128, 2000.

[7] C. Zhang, R. Lu, X. Lin, P.H. Ho, X. Shen, "An Efficient Identity-Based Batch Verification Scheme for Vehicular Sensor Networks", *Proceedings of the IEEE INFOCOM*, pp. 816–824, April 2008.

[8] A. Wasef, X. Shen, "PPGCV: Privacy Preserving Group Communications Protocol for Vehicular Ad Hoc Networks", *Proceedings of the IEEE ICC,* pp. 1458–1463, May 2008.

[9] H. Wen, P.H. Ho, G. Gong, "A Novel Framework for Message Authentication in vehicular Communication Network", *Proceedings of the IEEE GLOBECOM*, pp. 1-6, December 2009.

[10] A. Wasef, X. Shen, "MAAC: Message Authentication Acceleration Protocol for Vehicular Ad Hoc Networks", *Proceedings of the IEEE GLOBECOM*, pp. 1–6, December 2009.

[11] B.K. Chaurasia, S. Verma, S.M. Bhasker, "Message Broadcast in VANETs using Group Signature", *Proceedings of the IEEE WCSN*, pp. 131–136, December 2008.

[12] Y. Hao, Y. Cheng, K. Ren, "Distributed Key Management with Protection against RSU Compromise in Group Signature based VANETs", *Proceedings of the IEEE GLOBECOM*, pp. 1–5, December 2008.

[13] A. Studer, E. Shi, F. Bai, A. Perrig, "Together Efficient Authentication, Revocation, and Privacy in VANETs", *Proceedings of the IEEE SECON*, pp. 1–9, June 2009.

[14] R. Lu, X. Lin, H. Zhu, X. Shen, "SPARK: A New VANET Based Smart Parking Scheme for Large Parking Lots", *Proceedings of the IEEE INFOCOM*, pp. 1413–1421, April 2009.

[15] M. Bellare, P. Rogaway, "Random Oracles are Partial: A Paradigm for Designing Efficient Protocols", *Proceedings of the CCS,* pp. 67-73, 1993.