# CREDIT COIN: A PRIVACY-PRESERVING BLOCKCHAIN-BASED INCENTIVE ANNOUNCEMENT NETWORK FOR COMMUNICATION OF SMART VEHICLES

Submitted in partial fulfillment of the requirements for the award of Bachelor Degree in Computer science

By

**Naveen Kumar N ( Reg.No:38290055 )**

**Naresh Kumar J ( Reg.No:38290054 )**

**DEPARTMENT OF COMPUTER SCIENCE**

**SCHOOL OFCOMPUTING**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade "A" by NAAC**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI-600119

**APRIL 2021**

# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

**(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI,CHENNNAI-600 119

.**DEPARTMENT OF COMPUTER SCIENCE**

## BONAFIDE CERTIFICATE

This is to certify that the Project Report is the bonafide work of Mr. Naveen Kumar N ( Reg.No:38290055 ) and Mr. Naresh Kumar J ( Reg.No:38290054 ) who carried out the project entitled" CREDIT COIN: A PRIVACY-PRESERVING BLOCKCHAIN-BASED INCENTIVE ANNOUNCEMENT NETWORK FOR COMMUNICATION OF SMART VEHICLES" under our Supervisor from November 2020 to April 2021

**Internal guide**

**Mrs.REFONAA ,M.E., (Ph.D)**

**Head of the department**

**Dr.S..VIGNESHWARI,M.E.,Ph.D.,**

**Submitted for Viva voice Examination held on**

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## DECLARATION

We Naveen Kumar N ( Reg.No:38290055 ) and Naresh Kumar J ( Reg.No:38290054 ) hereby declare that the Project Report entitled " CREDIT COIN: A PRIVACY-PRESERVING BLOCKCHAIN-BASED INCENTIVE ANNOUNCEMENT NETWORK FOR COMMUNICATION OF SMART VEHICLE" done by us under the guidance of Mrs. REFONAA at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of science in computer science

**DATE:**

**PLACE:** **SIGNATURE OF THE CANDIDATE**

# ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **BOARD OF MANAGEMENT OF SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.SASIKALA, M.E.,Ph.D.,Dean** school of computer science and engineering and **Dr.S.Vigneshwari, M.E., Ph.D.and Dr.L.Lakshmanan M.E., Ph.D.,** Head of the Department, Department of Computer Science for providing me the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.Refonaa, M.E,Ph.D.,** for her valuable guidance, suggestions which paved way for the successful completion of my project work.

I wish to express my thanks to all teaching and non-teaching staff members of the **DEPARTMENT OF COMPUTER SCIENCE** who were helpful in many ways for the completion of project.

## ABSTRACT

With the increasing privacy concerns in VANETs, since the messages should be forwarded anonymously in VANETs, several attacks (e.g., the Sybil attack) have drawn a lot of attention. These attacks lead to a trade- off between users' privacy and message's reliability. Thus, some issues of privacy, such as anonymity, reliability, link ability (i.e., two signatures on the same message by one signer could be linkable) and traceability have become the main topics to be studied. Kounga et al. proposed a secure hardware mechanism to control the generation of pseudonyms for preventing Sybil attacks. Wu et al. used one-time authentication and message-linkable group signatures to identify malicious users. However, the trace phase requires expensive pairing operations so that it is inefficient to trace doubtable messages. Chen et al. proposed a threshold anonymous announcement (i.e., TA- Announcement) scheme with direct anonymous attestation and one-time anonymous authentication. In their scheme, the credentials of the malicious users cannot be revoked efficiently, and thus frequent attacks from malicious vehicles would decrease the efficiency of the scheme.

Vehicular announcement network in VANETs (Vehicular ad hoc networks) have become one of the most promising vehicular communication applications, as it leads to a much safer vehicle-driving experience. Block chain-based networks are promising in recording credit data with the good properties of tamper resistance and decentralization, which is useful in VANET's. We propose a vehicular announcement protocol Echo-Announcement in Credit Coin. It achieves efficiency and privacy-preserving for the practical usage in forwarding announcements. We design an incentive mechanism based on Block chain in Credit Coin. Users manage reputation points while they earn or spend coins as incentives. Meanwhile, Credit Coin still preserves privacy and achieves anonymity. Moreover, based on Block chain, Credit Coin prevents many security attacks and achieves conditional privacy because Trace manager will trace malicious nodes when an unexpected event occurs.

# TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| S.NO | ABBREVIATION | EXPANSION |
|---|---|---|
| 1 | JDK | AVA DEVELOPMENT TOOLKIT |
| 2 | DEX | DALVIK EXECUTABLES |
| 3 | TCP | TRANSMISSION CONTROL PROTOCOL |
| 4 | IP | INTERNET PROTOCOL |
| 5 | HTTP | HYPER TEXT TRANSFER PROTOCOL |
| 6 | ADT | ANDROID DEVELOPMENT TOOL |

# CHAPTER 1

# INTRODUCTION

## Aim:

The main aim of this project is to develop an effective announcement network called Credit Coin, a new privacy-preserving incentive announcement network based on Block chain.

## Synopsis:

Credit network is a common method to describe the credit relations among users in the network. In Credit networks, each node has points related to reputation, and it is easy to identify whether a node is honest or malicious by judging the reputation points. Therefore, it is widely applied in the digital currency of decentralized networks and Sybil-tolerant systems. Recently, Kate considered building a Blockchain-based Credit network in anonymous and Sybil-tolerant networks. Blockchain is currently idely studied on cryptocurrency in recent years. Nakamoto proposed Bicoin, which is a decentralized cryptocurrency based on Blockchain.

Bitcoin is popular and claimed as a kind of anonymity currency. However, due to the property of de- centralization, it can obtain the relations between different addresses by tracing a series of transactions. Therefore, there exists related work focusing on studying Blockchain-based networks in a privacy-preserving manner.

# CHAPTER 2
# LITERATURE SURVEY

Project Title: TrustDavis: A Non-Exploitable Online Reputation system
Author Name: Dimitri do B. DeFigueiredo Earl T. Barr Year of Publishing:
Abstract:

We present TrustDavis, an online reputation system that provides insurance against trade fraud by leveraging existing relationships between players, such as the ones present in social networks. Using TrustDavis and a simple strategy, an honest player can set an upper bound on the losses caused by any malicious collusion of players. In addition, TrustDavis incents participants to accurately rate each other, resists participants' pseudonym changes, and is inherently distributed.

Project Title: A Survey on Vehicular Ad hoc Networks Author Name: Mr. Bhagirath Patel, Ms. Khushbu Shah Year of Publishing: Dec. 2013
Abstract:

Vehicular Ad hoc Networks (VANETs), a subclass of mobile ad hoc network (MANET), is a promising approach for the intelligent transport system (ITS). VANET allows vehicles to form a self-organized network without the need for a permanent infrastructure. As the VANET has a potential in improving road safety, real time traffic update and other travel comforts, it turns attention of the researcher. Though VANET and MANET shares some common characteristics like self-organized network, dynamic topology, ad hoc nature etc, VANET differs from MANET by challenges, application, architecture, power constraint and mobility patterns, so routing protocols used in MANET are not applicable with VANET. New routing strategy for VANET has been proposed by many researchers in recent year. This paper provides focus on the various aspects of VANET like architecture, characteristic, challenges, glimpse of routing protocols, and simulation models used for VANET.

Project Title: Aggregation of Trustworthy Announcement Messages in Vehicular Ad Hoc Networks

Author Name: Alexandre Viejo, Francesc Seb Josep Domingo-Ferrer
Year of Publishing: 2009 Abstract:

Vehicular ad hoc networks (VANETs) allow vehicle to- vehicle communication and, in particular, vehicle- generated announcements. Vehicles can use such announcements to warn vehicles about road conditions (traffic jams, accidents). Thus, they can greatly increase the safety of driving. However, their trustworthiness must be guaranteed. A new system for vehicle-generated announcements is presented that is secure against external and internal attackers attempting to send fake messages. Internal attacks are thwarted by using an endorsement mechanism based on multi signatures. Besides, this scheme ensures that vehicles volunteering to generate and/or endorse trustworthy announcements do not have to sacrifice their privacy.

Project Title: Zerocash: Decentralized Anonymous Payments from Bitcoin
Author Name: Eli Ben-Sasson_, Alessandro Chiesay, Christina Garmanz, Matthew Greenz, Ian Miersz, Eran Tromerx, Madars Virzay

Year of Publishing: 2014 Abstract:
Bitcoin is the first digital currency to see widespread adoption. While payments are conducted between pseudonyms, Bitcoin cannot offer strong privacy guarantees: payment transactions are recorded in a public

decentralized ledger, from which much information can be deduced. Zerocoin (Miers et al., IEEE S&P 2013) tackles some of these privacy issues by unlinking transactions from the payment's origin. Yet, it still reveals payments' destinations and amounts, and is limited in functionality. In this paper, we construct a full-fledged ledger-based digital currency with strong privacy guarantees. Our results leverage recent advances in zero-knowledge Succinct Non-interactive Arguments of Knowledge (zk-SNARKs). First, we formulate and construct decentralized anonymous payment schemes (DAP schemes). A DAP scheme enables users to directly pay each other privately: the corresponding transaction hides the payment's origin, destination, and transferred amount. We provide formal definitions and proofs of the construction's security. Second, we build Zerocash, a practical instantiation of our DAP scheme construction. In Zerocash, transactions are less than 1 kB and take under 6 ms to verify — orders of magnitude more efficient than the less-anonymous Zerocoin and competitive with plain Bitcoin.

Project Title: Efficient Secure Aggregation in VANETs

Author Name: Maxim Raya, Adel Aziz and Jean-Pierre Hubaux

Year of Publishing: 2006

Abstract:

In VANETs, better communication efficiency can be achieved by sacrificing security and vice versa. But VANETs cannot get started without either of them. In this paper, we propose a set of mechanisms that can actually reconcile these two contradictory requirements. The main idea is to use message aggregation and group communication. The first class solutions are based on asymmetric cryptographic primitives, the second class uses symmetric ones, and the third onemixes the two. We have also evaluated the performance potential of one technique and arrived at the conclusion that aggregation in VANETs increases not only efficiency but also security.

Project Title: Fast and Secure Multi-hop Broadcast Solutions for Inter-Vehicular Communication

Author Name:  Wafa Ben Jaballah, Mauro Contiy, Mohamed Mosbah, Claudio E. Palazzi

Year of Publishing: 2012

Abstract:

Inter-vehicular communication (IVC) is an important emerging research area that is expected to contribute considerably to traffic safety and efficiency. In this context, many possible IVC applications share the common need for fast multi hop message propagation, including information such as position, direction, speed, etc. Yet, it is crucial for such data exchange system to be resilient to security attacks. Conversely, a malicious vehicle might inject incorrect information into the inter- vehicle wireless links leading to life and money losses, or to any other sort of adversarial selfishness (e.g., traffic redirection for the adversarial's benefit). In this work we analyze attacks to the state of the art IVC based safety applications. Furthermore, this analysis leads us to design a Fast and Secure Multi-hop BroadcastAlgorithm (FS-MBA) for vehicular communication, which results resilient to the aforementioned attacks.

Project Title: Privacy Issues in Vehicular Ad Hoc Networks

Author Name: Florian Dotzer Year of Publishing:

Abstract:

Vehicular Ad hoc Networks (VANETs) demand a thorough investigation of privacy related issues. On one hand, users of such networks have to be prevented from misuse of their private data by authorities, from location profiling and from other attacks on their privacy. On the other hand, system operators and car manufacturers have to be able to identify malfunctioning units for sake of system availability and security. These requirements demand an architecture that can really manage privacy instead of either providing full anonymity or no privacy at all. In this paper we give an overview on the privacy issues in vehicular ad hoc networks from a car manufacturer's perspective and introduce an exemplary approach to overcome these issues.

Project Title: Proving Reliability of Anonymous Information in VANETs
Author Name: Liqun Chen, Siaw-Lynn Ng Year of Publishing: 2010
Abstract:

Three vehicle-to-vehicle communication schemes by Kounga et al. ("Proving reliability of anonymous information in VANETs," IEEE Trans. Veh. Technol., vol. 58, no. 6, pp. 2977–2989, Jul. 2009) were recently published to address the issues of certificate management, scalability, and privacy. We present a number of attacks on one of the schemes. Our result shows that, contrary to what is claimed, this scheme does not provide the following four security features: 1) authenticity of a message; 2) privacy of drivers and vehicles; 3) reliability of distributed information; and 4) revocation of illegitimate vehicles.

Project Title: Reaching Agreement in the Presence of Faults
Author Name: M. PEASE, R, SHOSTAK, AND L. LAMPORT
Year of Publishing:
Abstract:

The problem addressed here concerns a set of isolated processors, some unknown subset of which may be faulty, that communicate only by means of two-party messages. Each non faulty processor has a private value of reformation that must be communicated to eachother non faulty processor. Non faulty processors always communicate honestly, whereas faulty processors may lie The problem is to devise an algorithm in which processors communicate their own values and relay values received from others that allows each non faulty processor to refer a value for each other processor The value referred for a non faulty processor must be that processor's private value, and the value inferred for a faulty one must be consistent wRh the corresponding value inferred by each other non faulty processor It is shown that the problem is solvable for, and only for, n >_ 3m + 1, where m IS the number of faulty processors and n is the total number. It is also shown that if faulty processors can refuse to pass on reformation but cannot falsely relay information, the problem is solvable for arbitrary n _> m _> 0. This weaker assumption can be approximated m practice using cryptographic methods

Project Title: Threshold anonymous announcement in VANETs
Author Name: Guilin Wang, L Chen, S. L. Ng
Year of Publishing: 2011 Abstract:

Vehicular ad hoc networks (VANETs) allow wireless communications between vehicles without the aid of a central server. Reliable exchanges of information about road and traffic conditions allow a safer and more comfortable travelling environment. However, such profusion of information may allow unscrupulous parties to violate user privacy. On the other hand, a degree of audit ability is desired for law enforcement and maintenance purposes. In this paper we propose a Threshold Anonymous Announcement service using direct anonymous attestation and one-time anonymous authentication to simultaneously achieve the seemingly contradictory goals of reliability, privacy and audit ability.

**MERITS AND DEMERITS:**

1. **Trust Davis: A Non-Exploitable Online Reputation System**

**Advantages:**

- Honest participants can limit the damage caused by *malicious* collusions of dishonest participants. Malicious participants gain no significant advantage by changing or issuing themselves multiple identities.
- There is strong incentive for participants to provide *accurate* ratings of each other. It requires no centralized services, and thus can be easily distributed.

**DisAdv:**

- The reputation systems now available on the internet can be manipulated by malicious individuals or groups for selfish purposes.

2. **A Survey On Vehicle Ad-Hoc Networks**

**Advantages:**

- We presented basic fundamentals of VANETs like architecture, characteristics, challenges and fundamental of routing and various types of routing VANET.

**DisAdv:**

- We presented basic fundamentals of VANETs like architecture, characteristics, challenges and fundamental of routing and various types of routing VANET.

3. **Aggregation Of Trustworthy Announcement Messages In Vehicle Ad Hoc Networks**

**Advantages:**

- A new system has been presented for trustworthy vehicle-generated announcements on VANETs that relies on a priori measures against internal attackers (vehicles in the VANET sending fake messages).
- System uses multi signatures over a Gap Diffie-Hellman group to aggregate announcements and reduce communication overhead

**DisAdv:**

- Vehicles trustworthiness must be guaranteed.

4. **Zerocash: Decentralized Anonymous Payments From Bitcoin**

**Advantages:**

- Zerocoin tackles some of these privacy issues by unlinking transactions from the payment's origin.

- We construct a full-fledged ledger-based digital currency with strong privacy guarantees

**DisAdv:**

- To protect privacy, users need an instant, risk-free, and, most importantly, automatic guarantee that data revealing their spending habits and account balances is not publicly accessible by their neighbors, co-workers, and merchants.

5. **Efficient Secure Aggregation In VANETs**

**Advantages:**

- Proposed several mechanisms, including combined signatures, overlapping groups, and dynamic group key creation.
- Addressed the tradeoff between efficiency and security in VANETs

**DisAdv:**

- Most VANET application designers attempt to minimize costs, sometimes even suggesting to scrap security totally.

6. **Fast And Secure Multi-Hop Broadcast Solutions For Inter- Vehicular Communication**

**Advantages:**

- The main goal of Inter Vehicular Communications (IVC) consists in increasing people's safety by exchanging warning messages between vehicles.
- Elaborated on security issues in IVC considering a general class of applications based on multi-hop broadcast; yet, without loss of generality.

**DisAdv:**

- It is crucial for data exchange system to be resilient to security attacks.

Conversely, a malicious vehicle might inject incorrect information into the inter- vehicle wireless links leading to life and money losses, or to any other sort of adversarial selfishness (e.g., traffic redirection for the adversarial benefit).

## 7. Privacy Issues In Vehicular Ad Hoc Network Advantages:

- Proposed a possible solution, based on prototypical experiments that we made and discussed its strengths and weaknesses.
- Discussed some threats to privacy in VANETs and argued why privacy is important. We also pointed out that the degree of privacy depends on user preferences, environmental settings, and application requirements and should therefore be adjustable.

**DisAdv:**

- Vehicular Ad hoc NETworks (VANETs) demand a thorough investigation of privacy related issues.

## 8. Threshold Anonymous Announcement In VANETs

**Advantages:**

- Presented a novel Threshold Anonymous Announcement scheme for VANET communication based on direct anonymous attestation and one-time anonymous authentication.
- Resolves the issues of non-repudiation and distinguish ability of origin which were previously unresolved, thereby enabling a reliable and auditable TAA service while preserving user privacy against both authorized parties and adversaries.

**DisAdv:**

- Exchanges of information about road and traffic conditions, if reliable, would enable a safer and more conducive travelling environment. On the other hand, such profusion of information may allow unscrupulous parties to track vehicles for profiling or more invidious purposes.

# CHAPTER 3

# PROJECT PURPOSE AND SCOPE

## Purpose

The goal of our work is to design an effective vehicular announcement network for VANETs. Based on the proposed incentive mechanism and Echo- Announcement, CreditCoin has the following properties: **Enthusiasm:**CreditCoin motivates users with incentives to share traffic information via announcements. It is the vehicular incentive announcement network in VANETs. **Privacy:**The requests, announcements, and the transactions do not leak any information about their sources (anonymity). Two messages in CreditCoin cannot be linked to the same sources (unlinkability). Only the TM reveals the user's identity when a un- expectancy occurs (traceability in conditional privacy).

**Reliability:** The announcements are signed by several honest witnesses (truthfulness). According to threshold authentication and Blockchain, every user could manage a copy of the whole block chains of transactions, and each transaction is related to the phases of announcement aggregation. Therefore, a source is unable to deny sending messages (non-reputation).

Additionally, announcements and transactions cannot be modified without authorization (tamper-resistance).

With the increasing privacy concerns of data, there exist two major issues in building an effective vehicular announcement network. First, ideally, all messages must be forwarded anonymously in VANETs since they usually contain sensitive information of users, such as vehicle numbers, driving preferences and customer identities. However, forwarding messages anonymously does not assure the reliability of the messages, thus decreasing the credit of vehicular announcements.

Second, users usually lack enthusiasm to forward any messages in

VANETs if there is a risk that their privacy will be breached. In addition, users do not benefit from forwarding announcements, which also makes them lack motivation to respond to messages.

## Project Scope

In order to build an effective vehicular announcement network, there are two parts in CreditCoin. The first part is announcement protocol, namely Echo-Announcement. This protocol provides threshold authentication and a certain privacy level to guarantee that anonymous announcements are reliable in CreditCoin. Users set their roles as follows: An Initiator invites other witnesses as Repliers to agree with his/her announcement with corresponding signatures and generates an announcement with traffic information and responses signed by Repliers. Since there is a larger group of users concealing all of the participants in the protocol, the receivers of the announcement knows the number of participants but cannot figure out their identities.

The second part is Blockchain-based incentive mechanism that works together with Echo- Announcement. Every user in CreditCoin owns a credit account at several addresses. The account contains reputation points called the coins. Users reward traffic announcements from a certain area by paying some coins as incentives. They can also spend some coins to make an announcement for hunting others' reward missions. Thus, in CreditCoin, a user gets a small amount of coins from replying to the aggregation request for an announcement of others. Meanwhile, he/she also has a chance to hunt a large amount of coins by making an announcement to someone in particular, as someone else needs it.

## Product Perspective

In summary, we make the following contributions:

• To the best of our knowledge, **CreditCoin**is the first privacy-preserving Blockchain-based incentive network in VANETs. It is able to build trust in communications of smart vehicles.

• We propose a vehicular announcement protocol Echo-Announcement in **CreditCoin**. It achieves efficiency and privacy-preserving for the practical usage in forwarding announcements.

• We design an incentive mechanism based on Blockchain in **CreditCoin**. Users manage reputation points while they earn or spend **coins** as incentives. Meanwhile, **CreditCoin**still preserves privacy and achieves anonymity. Moreover, based on Blockchain, **CreditCoin**prevents many security attacks and achieves conditional privacy because Trace manager will trace malicious nodes when an unexpected event occurs.

• We implement **CreditCoin**systematically in the simulation of smart transportation in JavaFX 2.0 and Java Runtime Environment 1.8. The test results show that **CreditCoin**is efficient and practical in the simulations of the smart transportation and smart vehicles.

## System Features

1. **Consensus server:** The consensus server is an entity that receives transactions and participates in the consensus phase. RSUs or official public vehicles are the consensus servers in our **CreditCoin**. There are l servers in **CreditCoin**. Users are connected directly to at least one server. For each server sz , z = 1, 2, · · ·l, there is a Unique Node List (UNL), called UNLsz. The list records the identities of multiple servers, each of which is directly connected to the list. In the consensus phase, szonly believes the vote results sent by the server listed on its UNLsz. According to the proof of schwartzet al. in, when the probability that any of the servers in UNLszattempts to initiate a collusion with other servers in the same list is less than 20%, with the increasing number of servers, the probability of making an undesirable consensus approaches to none quickly. As

Armknechtetal.proved, in order to avoid bifurcation, the repetition rate of servers in two diffident UNLs equals to or is greater than ρ/2, where ρ is a threshold of a voting rate about yes.

2. **Cloud application server:** Cloud application server manages and stores non-privacy information in the VANETs, such as msgconsisting in AGPs. For security reasons, they are separated from the encrypted information to help the entire network operate safely. Application server spreads public information, such as missions and announcements. Cloud application server works as a watcher in **CreditCoin**.

3. **User (OBU):** The user is an entity that trades in **Credit Coin** network. He/she creates or receives transactions. A user behaves in varieties of roles, such as Hunter, Replier, Initiator, and Verifier. We will elaborate these roles later in the following part.

4. **Public role:** Public role is defined similarly to the user. However, it is more privileged than the user. It receives and sends transactions and creates **coins** as well.

5. **Trusted authority**: Trusted authority takes charge of the generation and delivery of public keys. It creates d addresses for each user, and records the relationship between users and addresses.

6. **Trace manager:** Trace manager is the role that traces malicious users. If a fraudulent transaction is reported to trace manager, Trace manager will trace the malicious users with the help of trusted authority and send a report to cloud application server.

**Design and Implementation Constraints**

**Constraints in Analysis**

- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts

- ♦ Constraints as a Separate Concept
- ♦ Constraints Implied by the Model Structure

**Constraints in Design**

- ♦ Determination of the Involved Classes
- ♦ Determination of the Involved Objects
- ♦ Determination of the Involved Actions
- ♦ Determination of the Require Clauses
- ♦ Global actions and Constraint Realization

**Constraints in Implementation**

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

## Other Nonfunctional Requirements

### Performance Requirements

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

### Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

# CHAPTER 4
# SYSTEM ANALYSIS

## EXISTING SYSTEM

In Existing Systems users usually lack enthusiasm to forward any messages in VANETs if there is a risk that their privacy will be breached. In addition, users do not benefit from forwarding announcements, which also makes them lack motivation to respond to messages. Ideally, all messages must be forwarded anonymously in VANETs since they usually contain sensitive information of users, such as vehicle numbers, driving preferences and customer identities. However, forwarding messages anonymously does not assure the reliability of the messages, and also suffer from heavy workload.

## PROPOSED SYSTEM

In proposed system we propose a new technique called Credit Coin. It achieves efficiency and privacy- preserving for the practical usage in forwarding announcements. We design an incentive mechanism based on Block chain in Credit Coin. Users manage reputation points while they earn or spend coins as incentives. Meanwhile, Credit Coin still preserves privacy and achieves anonymity. Moreover, based on Block chain, Credit Coin prevents many security attacks and achieves conditional privacy because Trace manager will trace malicious nodes when an unexpected event occurs.

# CHAPTER 5

# REQUIREMENT SPECIFICATIONS

## INTRODUCTION

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

## HARDWARE AND SOFTWARE SPECIFICATION

### HARDWARE REQUIREMENTS

- Operating System : Windows 7 and above (64-bit).
- Python : 3.6
- Java : Jdk 1.8

### SOFTWARE REQUIREMENTS

- ➢ Hard Disk : 500GB and Above
- ➢ RAM : 4GB and Above
- ➢ Processor : I3 and Above

### TECHNOLOGIES USED

- ➢ JAVA
- ➢ PYTHON

BLOCKCHAIN

JAVA

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

## INTRODUCTION TO JAVA

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name *7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

## WORKING OF JAVA

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of **main,** which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv[] of C), and is declared as a static method.

To output text from the program, we execute the **println**method of **System.out,** which is java's output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user’s program.

Java consists of two things :

- Programming language
- Platform

## THE JAVA PROGRAMMING LANGUAGE

Java is a high-level programming language that is all of the following:

- Simple
- Object-oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture-neutral
- Portable
- High-performance
- Multithreaded
- Dynamic

The code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both co implied and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works:

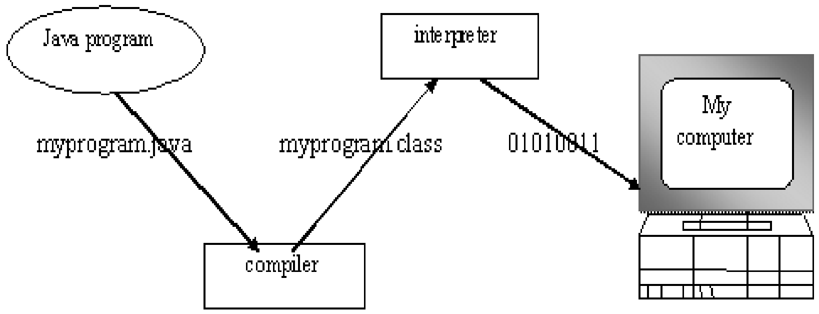
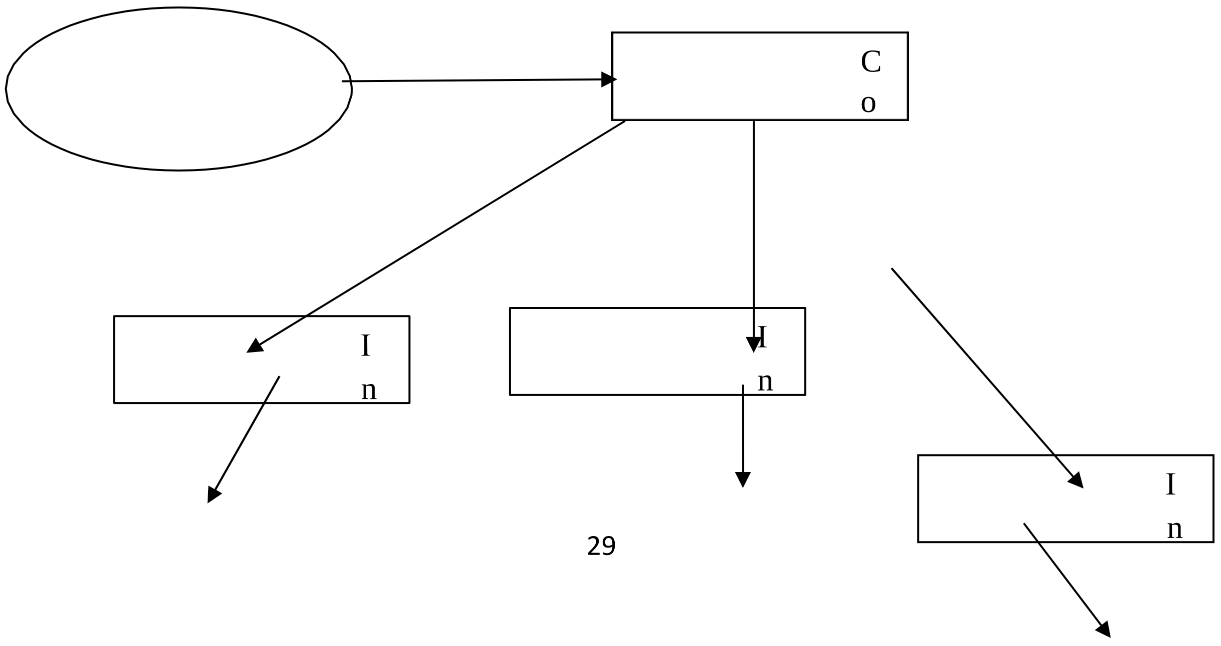
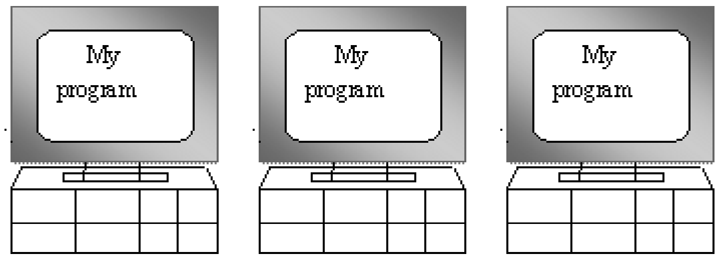


**Fig.5.1**

You can think of Java byte codes as the machine code instructions for the **Java Virtual Machine (JVM).** EveryJava interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible.

You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can e run on Windows NT, Solaris and Macintos





**PC-Compatible** **Sun Ultra Solaris Power macintosh**

**Windows NT System 8**

**THE JAVA PLATFORM**

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components :

- The Java Virtual Machine (JVM)
- The Java Application Programming Interface (Java API)

You've already been introduced to the JVM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries **(packages)** of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

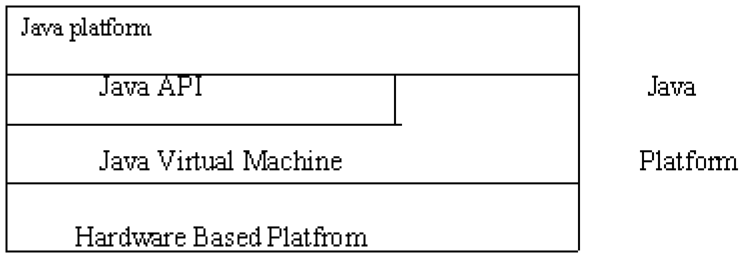


Fig.5.3

As a platform-independent environment, Java can be a bit slower than native code. However, smart compliers, weel-tuned interpreters, and just-in-time byte compilers can bring Java's performance close to that of native code without threatening portability.

**Introduction to Python**

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

**What can Python do?**

- Python can be used on a server to create web applications.

- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

**Why Python?**

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

**Good to know**

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.
- Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions

available are 2.7.15 and 3.6.5. However, an official End Of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.

- Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

**Python Syntax compared to other programming languages**

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**Python is Interpreted**

- Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
- This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

- One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
- For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
- Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

## Introduction to Blockchain

With the emergence of Digital Currency (aka Crypto currency), several enterprises or financial institutions are experimenting with the Distributed Ledger system as a trusted way to track the ownership of the assets without any central authority.

The core system behind the new currency system is Blockchain technology. A walkthrough of the basic building blocks of the Blockchain technology is described below.

A Blockchain is basically a chain of Blocks. Blocks are hashed using SHA-256 hashing algorithm to generate the signature of the data associated with it. Imagine a Blockchain as a linked-list whose node contains below attributes:

1. Block number – a sequence number (monotonically increasing) assigned to the block
2. Nonce – a random number which is used to generate Hash (as in #5) value which starts with 4 zeroes (0000). The process of generating this Nonce is called Mining.
3. Data – the actual user data associated with the block
4. Prev – contains the Hash of the previous block (e.g. current block # -1). The value for the first block in the chain is 64 zeroes (000000000000000000000000000000000000000000000000 0000000000000).
5. Hash – current block's Hash value (generated using SHA-256).

All of the above attributes excluding Hash e.g. Block #, Nonce, data, Prev are used to calculate the Hash of this block.

[#=1, Nonce=3409, Data=x, Prev=00..0, Hash=<u>0000ffgr5rg67j</u>]<- [#=2, Nonce=4986, Data=x, Prev=<u>0000ffgr5rg67j</u>, Hash=000045tggr5rg..77yh] <-

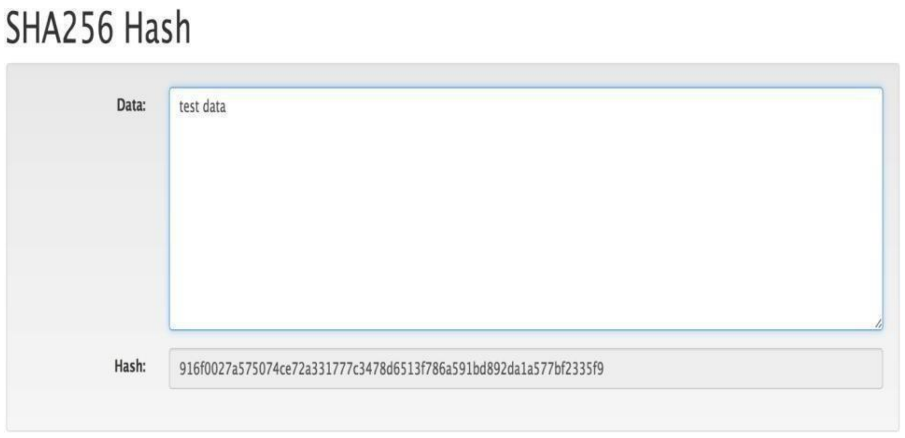
……and the chain goes on…

e.g. in above block #1, the value for Hash=0000ffgr5rg67j is generated using the values 1,3409,x,00..0. In case value for any of these 4 attributes changes, it will change the Hash value of this block. Once the Hash value of this Block changes (e.g. from 0000ffgr5rg67j to 34sdffgr5rg67j), it will break the next Block (#2) as its Prev field will point to invalid Hash (0000ffgr5rg67j doesn't exist anymore). This leads to a ripple effect and turns whole chain as invalid/tampered.

One way to fix it is to run mining and recalculate the Hash value of Block #1 which basically will generate new value for Nonce and hence leading to a valid Hash value which starts with 4 zeroes. Copying this to next Block #2's Prev field will fix these 2 Blocks. However in order to fix the whole Blockchain, we need to continue with this process for all the Blocks in the chain so that all Blocks point to new & valid Hash codes of their previous blocks.

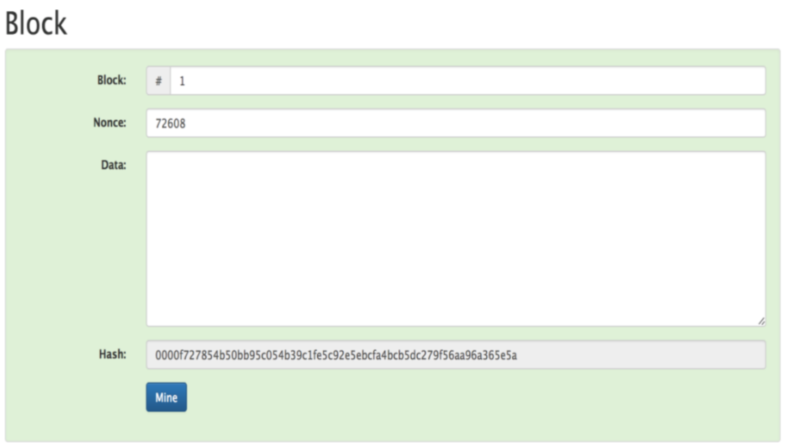
The cost of fixing the tampered Blockchain as described in above process is very high. Because we have to go and fix the Chain from the starting Block to the last one. In case the Chain is large, it becomes costly operation. In case of Distributed Blockchain where several Peers are involved in the process and keeping the copy of the Blockchain, the repairing the Blocks becomeeven more costly operation.

The other and more efficient process is to come up with the compensating data and add this Block at the end of the Chain. E.g. In case your Chain contains the financial transaction (money movement) in Data field of the Block, then instead of fixing each of the Block's Data with corrected financial transaction, come up with the adjusted financial transaction (aka compensating transaction) and create a Block (with Data=adjusted transaction record) and add this Block to the Blockchain (adds to the end of the Chain).


SHA256 Hash

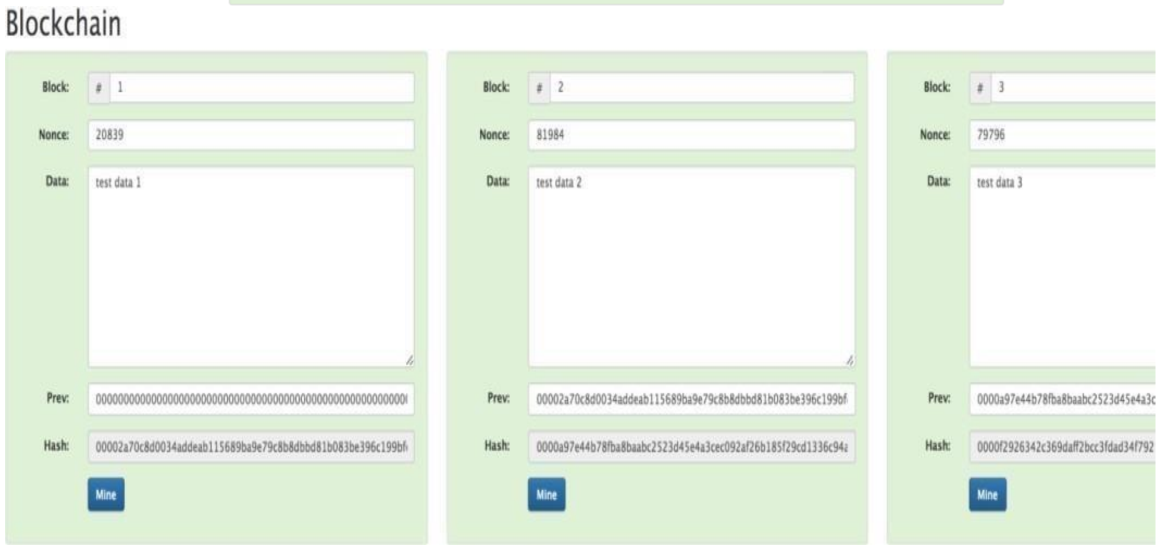
Data: test data

Hash: 916f0027a575074ce72a331777c3478d6513f786a591bd892da1a577bf2335f9



Block

| | |
|---|---|
| Block: | # 1 |
| Nonce: | 72608 |
| Data: | |
| Hash: | 0000f727854b50bb95c054b39c1fe5c92e5ebcfa4bcb5dc279f56aa96a365e5a |



Blockchain

| | | | |
|---|---|---|---|
| Block: | # 1 | # 2 | # 3 |
| Nonce: | 20839 | 81984 | 79796 |
| Data: | test data 1 | test data 2 | test data 3 |
| Prev: | 000000000000000000000000000000000000000000000000000000000 | 00002a70c8d0034addeab115689ba9e79c8b8dbbd81b083be396c199bf | 0000a97e44b78fba8baabc2523d45e4a3c |
| Hash: | 00002a70c8d0034addeab115689ba9e79c8b8dbbd81b083be396c199bf | 0000a97e44b78fba8baabc2523d45e4a3cec092af26b185f29cd1336c94a | 0000f2926342c369daff2bcc3fdad34f792 |

# CHAPTER 6

# SYSTEM DESIGN

## Architecture Diagram:

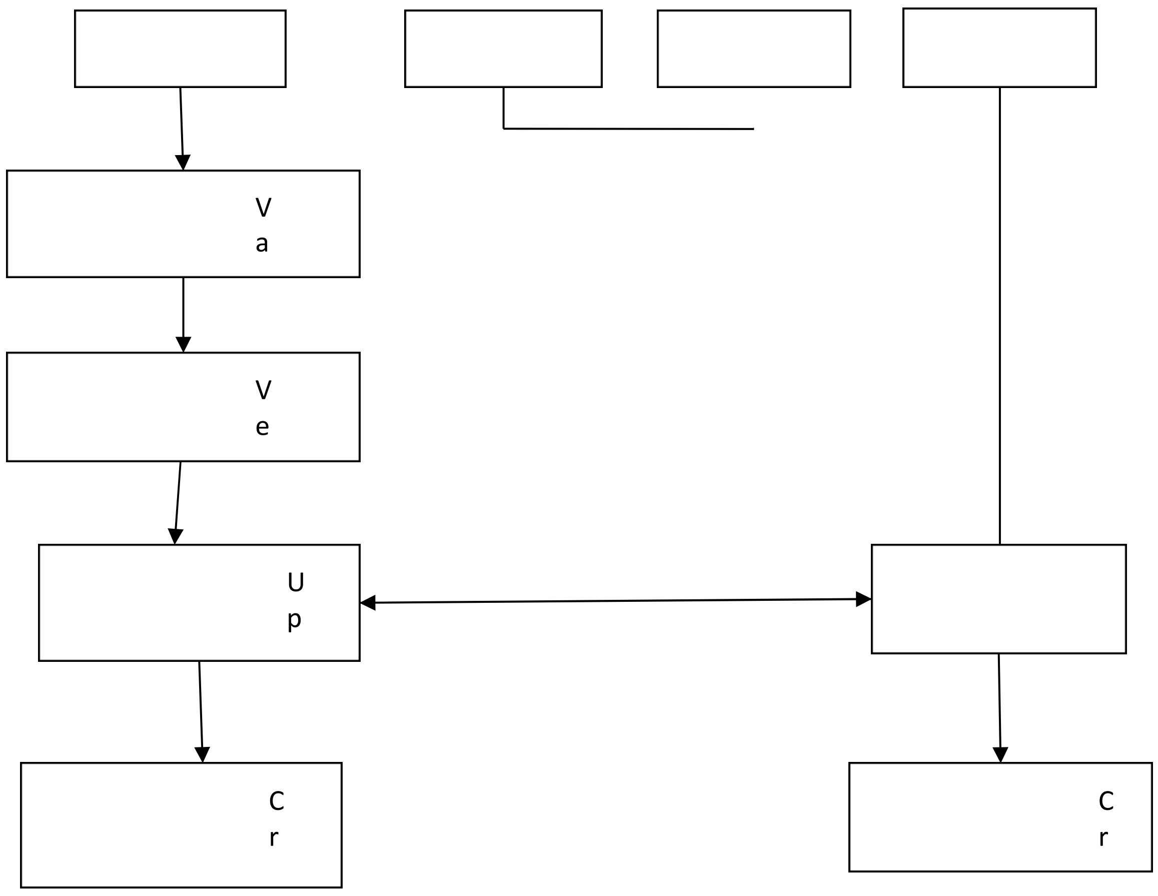
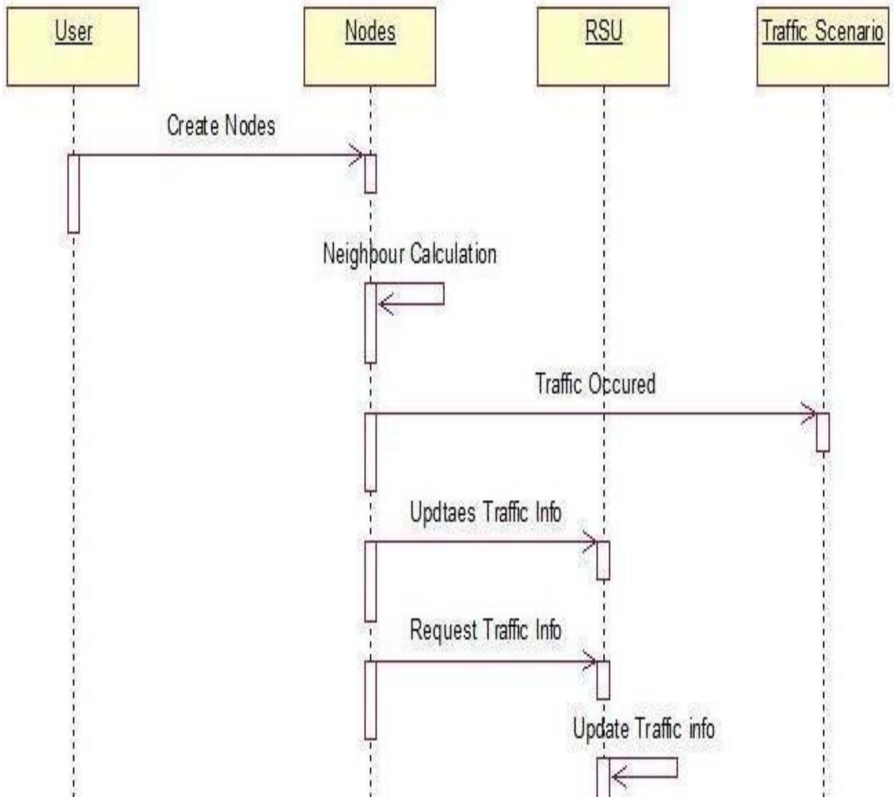


Fig: 6.1

## Sequence Diagram:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.


User
Nodes
RSU
Traffic Scenario
Create Nodes
Neighbour Calculation
Traffic Occured
Updtaes Traffic Info
Request Traffic Info
Update Traffic info

**Use Case Diagram:**

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

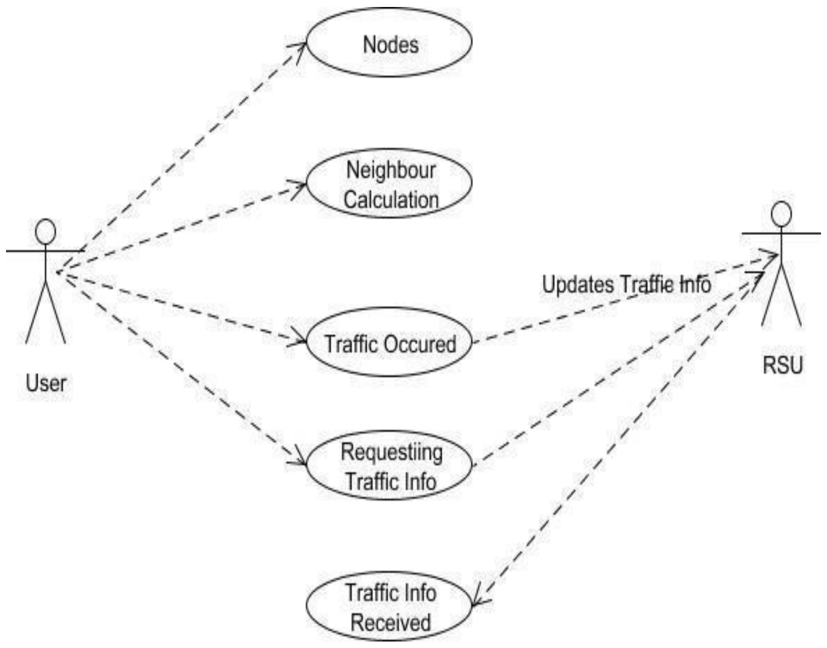
**USE CASE DIAGRAM**

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.

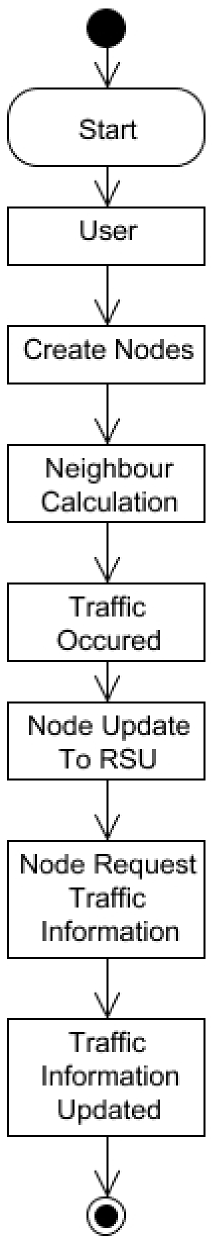


**Activity Diagram:**

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

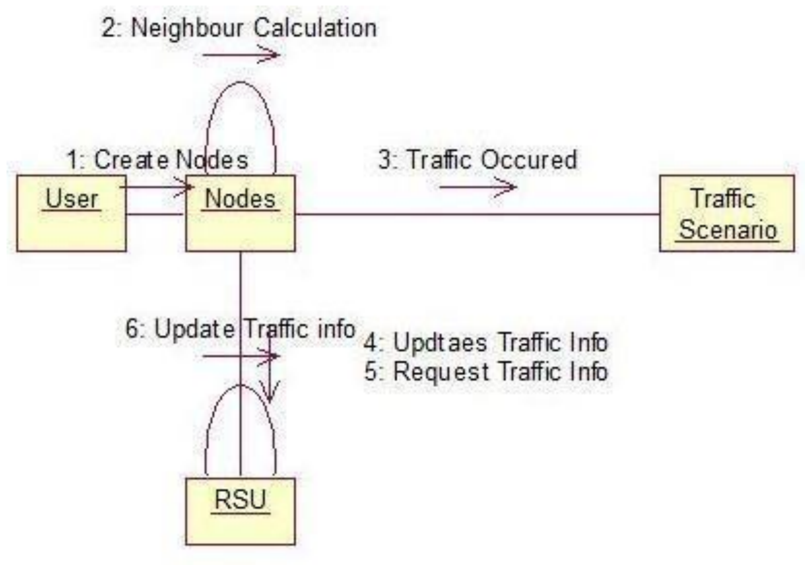
The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- Anencircled circle represents the end of the workflow.


Start
User
Create Nodes
Neighbour Calculation
Traffic Occured
Node Update To RSU
Node Request Traffic Information
Traffic Information Updated

**Collaboration Diagram:**

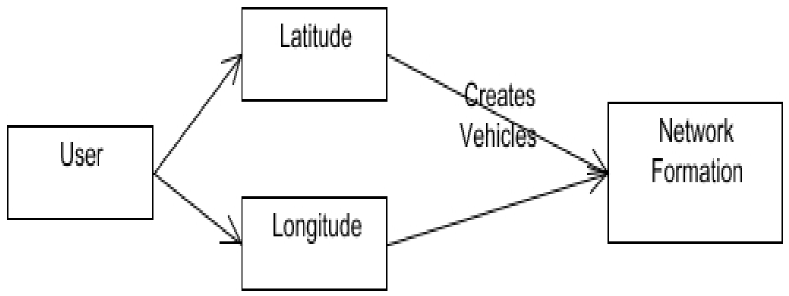
UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.



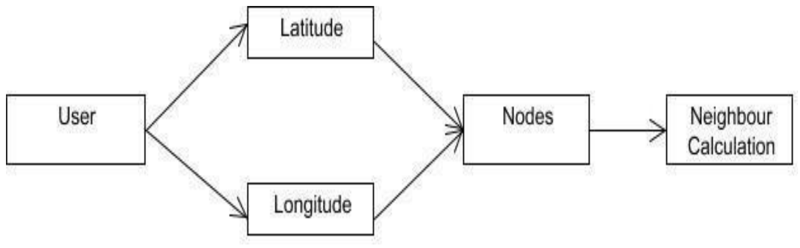
## DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

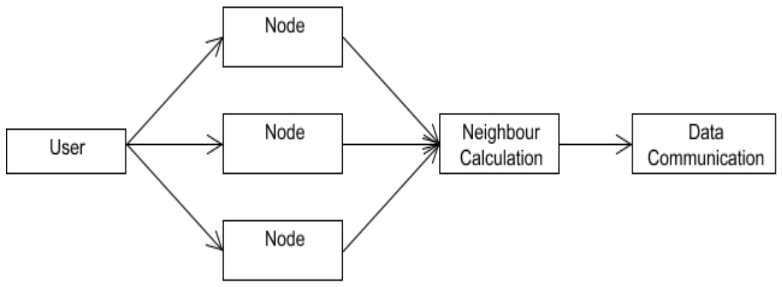
### Level 0:



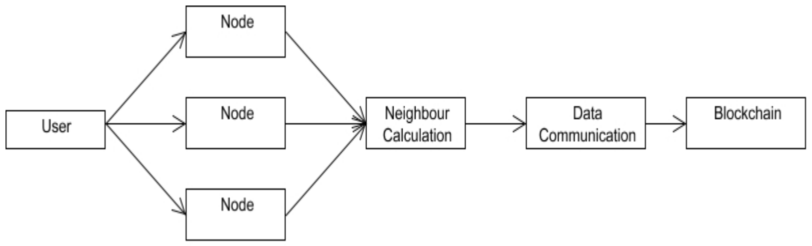
### Level 1:



### Level 2:

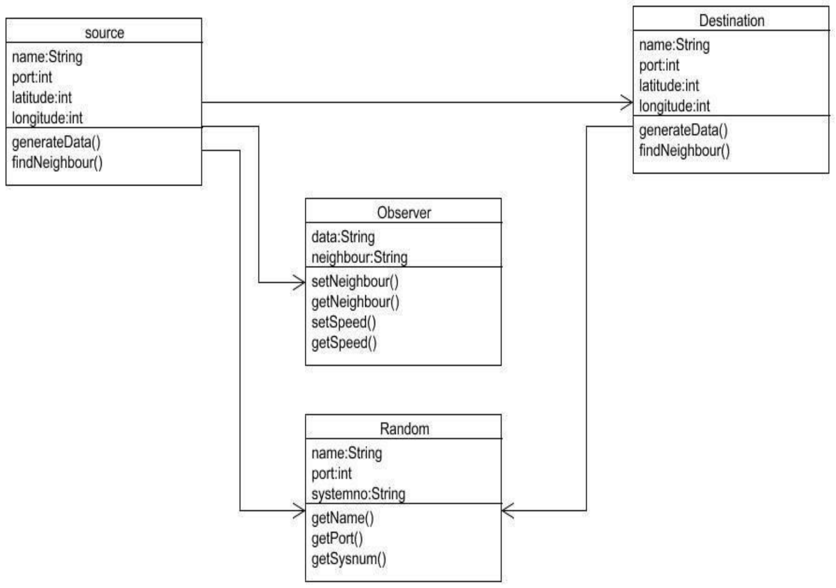


**Level 3:**



## Class Diagram

A Class diagram in the Unified Modelling Languageis a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



# CHAPTER 7

# SYSTEM DESIGN - DETAILED

Finally we design **CreditCoin**, a novel privacy- preserving Blockchain-based incentive announcement network with our vehicular announcement protocol Echo-Announcement in VANETs. Our announcement protocol maintains the reliability of announcements without revealing users' privacy and is reliable and efficient in the non-fully-trusted environment in VANETs.Furthermore, the designed incentive mechanism encourages users to be active in responding. With Blockchain, the security is also enhanced since announcements and transactions are traced only by Trace manager in **CreditCoin**. Through our simulations, the total time of transaction part for users is around 130ms per transaction, and the total time of consensus part for RSUs is around 92.4ms per 100 transactions. To conclude, **CreditCoin**is practical in the scenario of smart vehicles and smart transportation.

## MODULES

- ✓ **Network Formation**
- ✓ **Neighbor Calculation**
- ✓ **Data Communication**
- ✓ **Block Chain**

## MODULE EXPLANATION:

### Network Formation:

In this module, we create a network formation. A network formation consists of nodes. Each node has distance and range based on which coverage area is formed. Based on coverage area nodes communicate with each other and neighbor nodes are formed. If destination node is out of coverage area of source node, message transmitted to destination via neighbors.

## Neighbor Calculation:

After Network is formed based on vehicle location and road side unit location neighbor is calculated. In VANET environment vehicle to vehicle or vehicle to road side unit communication takes place based on their neighbors which have intersecting range. As vehiclesare dynamic neighbors are also dynamic and neighbors keep changing once vehicle starts to move from one location to another.

## Data Communication:

After Network is formed and neighbors are calculated dynamically data communication takes place between vehicle and road side unit. A vehicle in the network request another vehicle in the network about traffic status via road side unit as they were out of range.

## Block Chain:

The users request and messages were securely stored in Block chain implementation. When a credit coin were issued or received, it is considered a block and added to the block chain. The block chain uses the miners to append the transaction details as a block to the block chain.

# CHAPTER 8

# CODING AND TESTING

## CODING

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

## CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary.Some of the standard neededto achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand. Naming conventions
Value conventions

Script and comment procedure Message box format Exception and error handling

## NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self- descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

**Class names**

Class names are problem domain equivalence and begin with capital letter and have mixed cases

**Members function and Data Members name**

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

**VALUE CONVENTIONS**

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

**SCRIPT WRITING AND COMMENTING STANDARD**

Script writing is an art in which indentation is utmost important.

Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

**MESSAGE BOX FORMAT**

When something has to be prompted to the user, he must be able to understand it properly. To achieve this,a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

## TEST PROCEDURE SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

## TEST DATA AND OUTPUT

### UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

### FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

**PERFORMANCE TEST**

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

**STRESS TEST**

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

**STRUCTURED TEST**

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors

in output information

## INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in
combination. And so on. The advantages of thisapproach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

## TESTINGTECHNIQUES/TESTINGSTRATERGIES

### TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing isthe critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

**WHITE BOX TESTING**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

## BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

Graph based testing methods

- Equivalence partitioning
- Boundary value analysis
- Comparison testing

## SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

## INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

## PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

## SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

## VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

* The function or performance characteristics confirm to specifications and are accepted.
* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found tobe working satisfactorily. Though there were deficiencies in the system they were not catastrophic

## USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

# CHAPTER 9

# CONCLUSION AND FUTURE WORKS

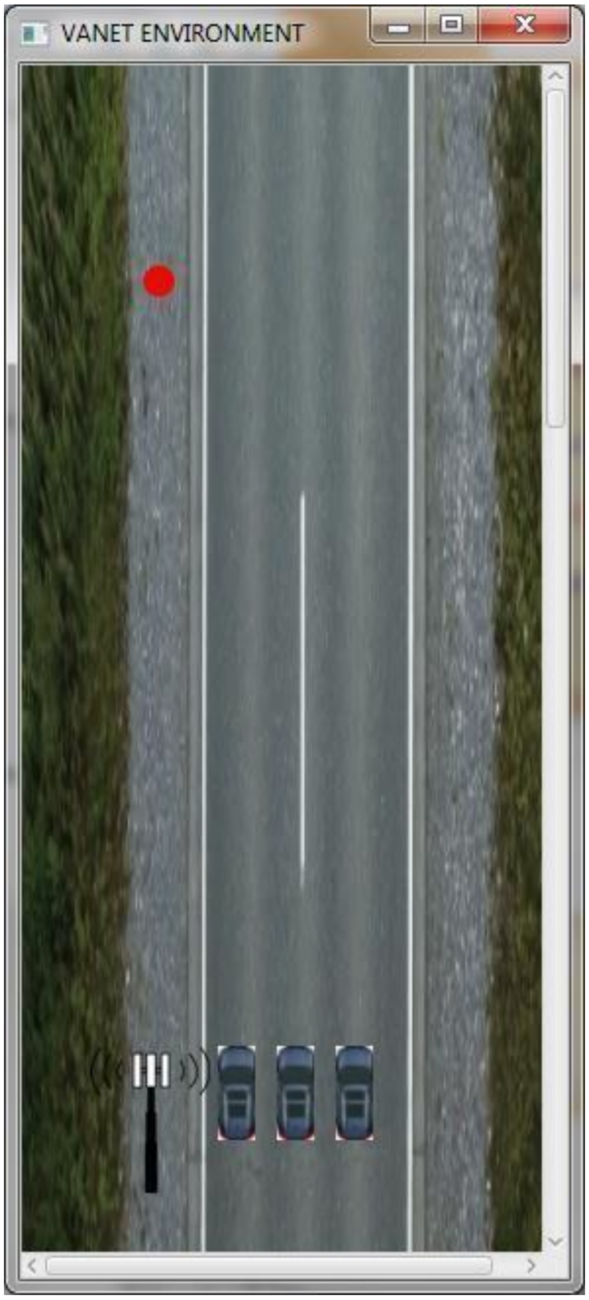
In this paper, we have proposed CreditCoin, a novel privacy-preserving Blockchain-based incentive announcement network with our vehicular announcement protocol Echo-Announcement in VANETs.

Our announcement protocol maintains the reliability of announcements without revealing users' privacy and is reliable and efficient in the non-fully-trusted environment in VANETs. Through our simulations, the total time of announcements for a user only is 174ms in our assumptions, which is much more efficient than other protocols. Furthermore , the designed incentive mechanism encourages users to be active in responding. With Blockchain, the security is also enhanced since announcements and transactions are traced only by Tracemanager in CreditCoin. Through our simulations, the total time of transcation part of users is around 130ms per transaction, and the total time of consensus part for RSUs is around 92.4ms per 100 transctions. To conclude, CreditCoin is practical in the scenario of smart vehicles and smart transportation.

## FUTURE WORK:

- In future work, we plan to improve the key management and the coin balance in CreditCoin.
- Designing more effective trading propositions is also being investigated. .

**RESULT:**



## REFERENCES

- [1] L. Chen, S.-L. Ng, and G. Wang, “Threshold anonymous announcement in VANETs,” IEEE J. Sel. Areas Commun., vol. 29, no. 3, pp. 605–615, Mar. 2011.
- [2] J. Shao, X. Lin, R. Lu, and C. Zuo, “A threshold anonymous authentication protocol for VANETs,” IEEE Trans. Veh. Technol., vol. 65, no. 3, pp. 1711–1720, Mar. 2016.
- [3] S. Nakamoto. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [Online]. Available: https://bitcoin.org/bitcoin.pdf
- [4] H. Hartenstein and L. P. Laberteaux, “A tutorial survey on vehicular ad hoc networks,” IEEE Commun. Mag., vol. 46, no. 6, pp. 164–171, Jun. 2008.
- [5] B. Parno and A. Perrig, “Challenges in securing vehicular networks,” in Proc. Workshop Hot Topics Netw. (HotNets-IV), MD, USA, Nov. 2005, pp. 1–6.
- [6] F. Dötzer, “Privacy issues in vehicular ad hoc networks,” in Proc. Int. Workshop Privacy Enhancing Technol., May 2005, pp. 197–209.
- [7] J. R. Douceur, “The sybil attack,” in Proc. Int. Workshop Peer- to-Peer Syst., 2002, pp. 251–260.
- [8] E. Bresson, J. Stern, and M. Szydlo, “Threshold ring signatures and applications to ad-hoc groups,” in Proc. Annu. Int. Cryptol. Conf.,Aug. 2002, pp. 465–480.
- [9] J. Ren and L. Harn, “An efficient threshold anonymous authentication scheme for privacy-preserving communications,” IEEE Trans. Wireless Commun., vol. 12, no. 3, pp. 1018–1025, Mar. 2013.
- [10] M. Raya, A. Aziz, and J.-P. Hubaux, “Efficient secure aggregation in VANETs,” in Proc. 3rd Int. Workshop Veh. Ad Hoc Netw., Sep. 2006, pp.

67–75.

- [11] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle,” in Proc. Black Hat USA, Aug. 2015, pp. 1-91.
- [12] R. G. Engoulou, M. Bellaïche, S. Pierre, and A. Quintero, “VANET security surveys,” Comput. Commun., vol. 44, pp. 1-13, May 2014.
- [13] B. Yu, C.-Z. Xu, and B. Xiao, “Detecting sybil attacks in VANETs,”J. Parallel Distrib. Comput., vol. 73, no. 6, pp. 746–756, Jun. 2013.
- [14] J. Petit and S. E. Shladover, “Potential cyberattacks on automated vehicles,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 546–556, Apr. 2015.
- [15] W. B. Jaballah, M. Conti, M. Mosbah, and C. E. Palazzi, “Fast and secure multihop broadcast solutions for intervehicular communication,” IEEE Trans. Intell. Transp. Syst., vol. 15, no. 1, pp. 433–450, Feb. 2014.
- [16] G. Kounga, T. Walter, and S. Lachmund, “Proving reliability of anonymous information in VANETs,” IEEE Trans. Veh. Technol., vol. 58, no. 6, pp. 2977–2989, Jul. 2009.
- [17] Q. Wu, J. Domingo-Ferrer, and Ú. Gonzalez-Nicolas, “Balanced trustworthiness, safety, and privacy in vehicle-to- vehicle communications,” IEEE Trans. Veh. Technol., vol. 59, no. 2, pp. 559–573, Feb. 2009.
- [18] B. Qin, Q. Wu, J. Domingo-Ferrer, and W. Susilo. (2012). Robust Distributed Privacy-Preserving Secure Aggregation in Vehicular Communication. [Online]. Available: http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1309&context=ei spapers
- [19] Y. Xia, W. Chen, X. Liu, L. Zhang, X. Li, and Y. Xiang, “Adaptive multimedia data forwarding for privacy preservation in

- vehicular adhoc networks," IEEE Trans. Intell. Transp. Syst., vol. 18, no. 10, pp. 2629–2641, Jan. 2017.
- [20] L. Zhang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "APPA: Aggregate privacy-preserving authentication in vehicular ad hoc networks," in Proc. Int. Conf. Inf. Secur., Oct. 2011, pp. 293–308.
- [21] X. Lin, "LSR: Mitigating zero-day sybil vulnerability in privacypreserving vehicular peer-to-peer networks," IEEE J. Sel. Areas Commun., vol. 31, no. 9, pp. 237–246, Sep. 2013.
- .[22] M. Azees, P. Vijayakumar, and L. J. Deboarh, "EAAP: Efficient anonymous authentication with conditional privacy- preserving scheme for vehicular ad hoc networks," IEEE Trans. Intell. Transp. Syst., vol. 18, no. 9, pp. 2467–2476, Sep. 2017.
- . [23] D. B. DeFigueiredo and E. T. Barr, "TrustDavis: A non- exploitable online reputation system," in Proc. IEEE Int. Conf. E- Commerce Technol. (CEC), Jul. 2005, pp. 274–283.
- [24] A. Ghosh, M. Mahdian, D. M. Reeves, D. M. Pennock, and
- R. Fugger, "Mechanism design on trust networks," in Proc. Int. Workshop Web Internet Econ., 2007, pp. 257–268.
- [25] D. Schwartz, N. Youngs, and A. Britto. (2014). The Ripple Protocol Consensus Algorithm, Ripple Labs Inc White Paper5. [Online]. Available: http://www.the-blockchain.com/docs/Ripple%20Consensus%20Whitepaper.pdf

## APPENDIX

**Source Code Creditcoin.java** packagecreditcoin;

```
importcom.controller.net.ApplicationBase;
importjavafx.application.Application; importjavafx.event.EventHandler;
importjavafx.fxml.FXMLLoader; importjavafx.scene.Parent;
importjavafx.scene.Scene; importjavafx.scene.input.MouseEvent;
importjavafx.stage.Stage; importjavafx.stage.StageStyle;
public class CreditCoin extends ApplicationBase { doubledragX=0;
doubledragY=0; @Override
public void start(Stage stage) throws Exception
{
    Parent root =
FXMLLoader.load(getClass().getResource("FXMLDocu ment.fxml"));

Scene scene = new Scene(root);
    String a=this.getClass().getResource("design.css").toExternalF orm();
scene.getStylesheets().add(a); stage.setScene(scene);
stage.initStyle(StageStyle.TRANSPARENT); stage.show();
root.setOnMouseDragged(new EventHandler<MouseEvent>()
{
@Override
public void handle (MouseEvent me)

    ▪

{
stage.setX(me.getScreenX() - dragX); stage.setY(me.getScreenY() -
dragY);
}
});
root.setOnMousePressed(new EventHandler<MouseEvent>()
{
@Override
public void handle (MouseEvent me)
{
dragX = me.getScreenX() - stage.getX(); dragY = me.getScreenY() -
stage.getY();
}
                                        });
}
public static void main(String[] args) { launch(args);
}
}
```

**Multisender.java** packagecreditcoin; importjava.net.DatagramPacket;

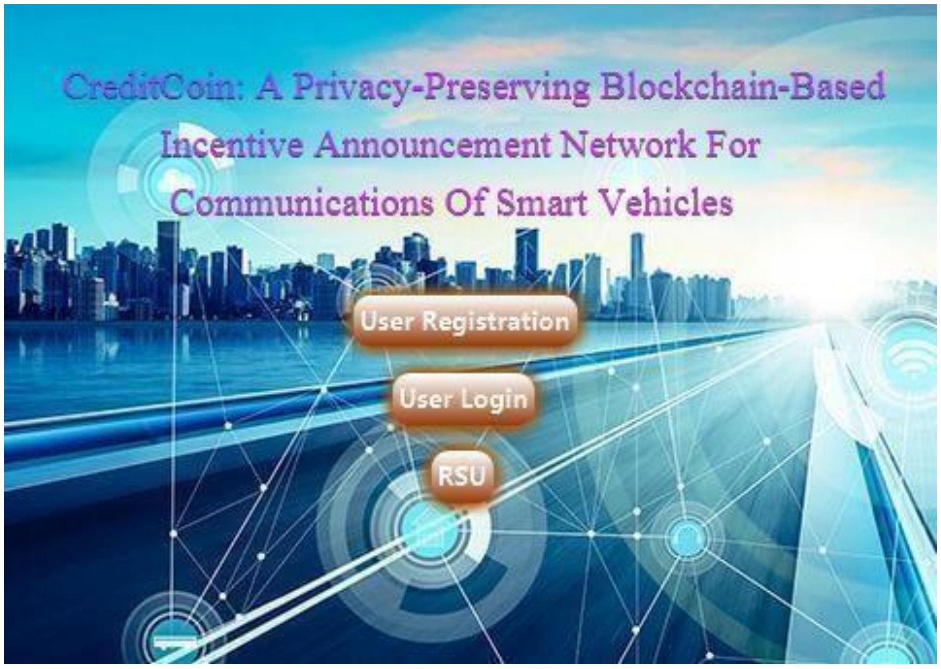
```
importjava.net.InetAddress; importjava.net.MulticastSocket;
public class MultiSender extends Thread
{
public String str,node,sys,port,xVal,yVal,signal; UserControllerucc;
RsuControllerrcc;
publicMultiSender(String
nodename,Stringportnumber,Stringsysnumber,StringxV
al,StringyVal,Stringsigna,UserControlleruc)
{
this.node=nodename; this.sys=sysnumber;
this.port=portnumber;
this.xVal=xVal; this.yVal=yVal; this.signal=signa;
this.ucc=uc;
str="CARINFO"+"$"+nodename+"$"+sysnumber
```

```
+"$"+po
rtnumber+"$"+xVal+"$"+yVal+"$"+signa;
start();
}
publicMultiSender(String
nodename,Stringportnumber,Stringsysnumber,StringxV
al,StringyVal,RsuControllerrsu)

{
this.node=nodename; this.port=portnumber;
      this.sys=sysnumber; this.xVal=xVal; this.yVal=yVal;
this.rcc=rsu;
str="RSUINFO"+"$"+nodename+"$"+portnumber+"$"+s
ysnumber+"$"+xVal+"$"+yVal;
start();
}
   @Override public void run()
{
try
{
InetAddress in = InetAddress.getByName("225.89.67.48");

MulticastSocket ms = new MulticastSocket(4567); ms.joinGroup(in);
DatagramPacketdp                                 =                     new
DatagramPacket(str.getBytes(), str.length(), in, 4567);

ms.send(dp); Thread.sleep(2000);

}
catch(Exception e)
{
e.printStackTrace();
}
}
}
```

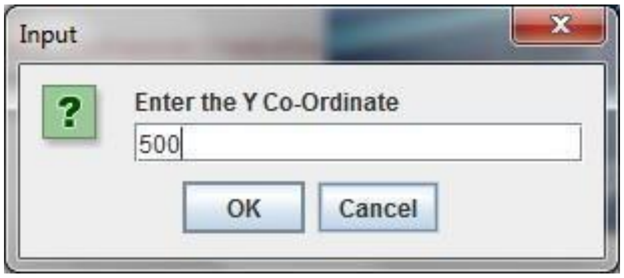
**Multireceiver.java** packagecreditcoin; importjava.io.ObjectOutputStream;

```
importjava.net.DatagramPacket; importjava.net.InetAddress;
importjava.net.MulticastSocket; importjava.net.Socket;
importjava.util.HashMap; importjava.util.StringTokenizer;
importjava.util.TreeMap; importjava.util.Vector;
public class MultiReceiver extends Thread{ public String node, sys, port,
xVal, yVal,signal;
    Observer o = new Observer(); UserControllercarc; RsuControllerrcc;
public static HashMapallcarsysnum = new HashMap(); public static
HashMapallcarportnum = new HashMap(); public static
HashMapallcarsignal=new HashMap();
public static HashMapallrsusysnum = new HashMap();
public static HashMapallrsuportnum = new HashMap();

publicMultiReceiver(String nodename, String portnumber, String
                                    sysnumber,Stringsigna, UserControlleruc)

{
this.node = nodename; this.port = portnumber;
      this.sys = sysnumber; this.signal=signa;
      o = uc.getObserver(); carc = uc;
start();
}
publicMultiReceiver(String nodename,Stringportnumber, String sysnumber,
RsuControllerrc)
{
this.node = nodename; this.port = portnumber;
this.sys = sysnumber; this.rcc=rc;
o=rc.getObserver();
rcc=rc; start();
}
  @Override public void run()
{
try
{
InetAddress                          in                    =
InetAddress.getByName("225.89.67.48");
MulticastSocket ms = new MulticastSocket(4567); ms.joinGroup(in);
while(true)
{
byte[] b = new byte[1024];

DatagramPacketdp = new DatagramPacket(b, b.length); ms.receive(dp);
           String               data1        =        new
String(dp.getData()).trim();
StringTokenizer str1 = new StringTokenizer(data1, "$"); String str2 =
str1.nextToken().toString();
if (str2.equals("CARINFO"))
{
String nodenum = str1.nextToken(); String sysnum = str1.nextToken();
String portnum = str1.nextToken(); String xVal=str1.nextToken(); String
yVal=str1.nextToken(); String signal=str1.nextToken();
allcarsignal.put(nodenum, signal); allcarsysnum.put(nodenum, sysnum);
allcarportnum.put(nodenum, portnum);
}
else if(str2.equals("RSUINFO"))
{
String rsuname=str1.nextToken(); String portnum=str1.nextToken(); String
sysnum=str1.nextToken();
allrsusysnum.put(rsuname, sysnum); allrsuportnum.put(rsuname, portnum);

}

}

}
catch(Exception e)
{
e.printStackTrace();
}
public          void    applyBreak(String    partialBreakNode, intxPos,
intyPos, int speed, String data)
{
Try
{
        String sysno =
allcarsysnum.get(partialBreakNode).toString().trim();
        String                  portno              =
allcarportnum.get(partialBreakNode).toString().trim();

        Socket          s     =     new     Socket(sysno,
Integer.parseInt(portno));
ObjectOutputStreamoos                   =              new
ObjectOutputStream(s.getOutputStream());

oos.writeObject("Break"); oos.writeObject(partialBreakNode);
oos.writeObject(xPos); oos.writeObject(yPos); oos.writeObject(speed);
oos.writeObject(data); oos.close();
s.close();
}
catch(Exception e)
{
e.printStackTrace();
}
}
public                void   criticalNeigh(String   neighName,   String
nodeName, int speed) {
try {
        String                                sysno                      =
allcarsysnum.get(neighName).toString().trim();

        String                                portno                     =
allcarportnum.get(neighName).toString().trim();

        Socket                 s        =        new        Socket(sysno,
Integer.parseInt(portno));
ObjectOutputStreamoos                                =                new
ObjectOutputStream(s.getOutputStream());

oos.writeObject("Critical"); oos.writeObject(neighName);
oos.writeObject(nodeName); oos.writeObject(speed); oos.close();
s.close();
      } catch (Exception e) { e.printStackTrace();
}
}
}
```

Screenshots:


CreditCoin: A Privacy-Preserving Blockchain-Based Incentive Announcement Network For Communications Of Smart Vehicles
User Registration
User Login
RSU

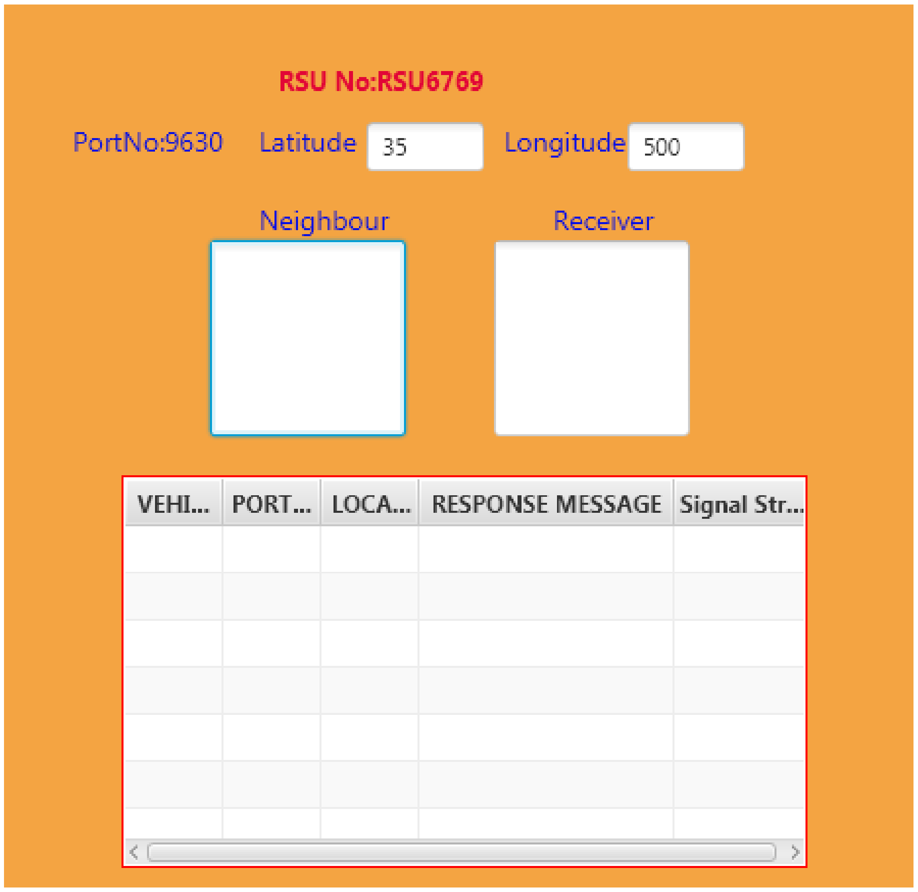
EnterRSUCo-Ordinate:



Input

Enter the Y Co-Ordinate

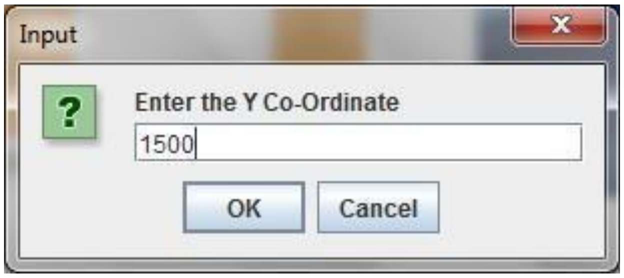
500



RSU No:RSU6769

PortNo:9630 Latitude 35 Longitude 500

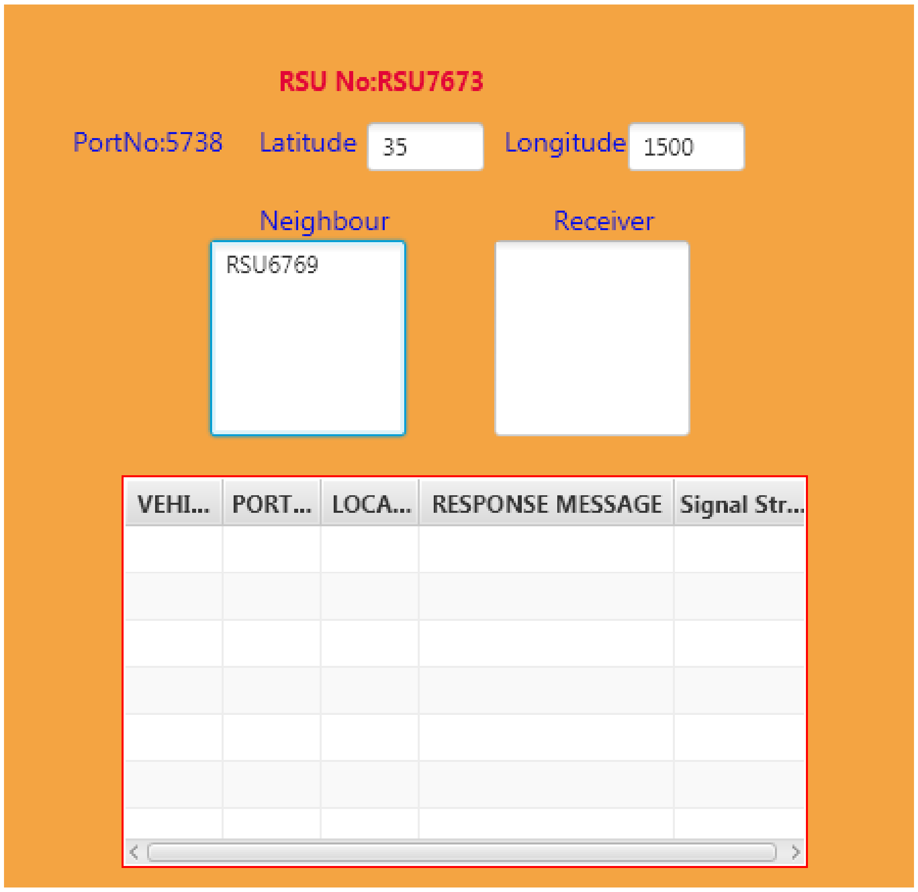
| VEHI... | PORT... | LOCA... | RESPONSE MESSAGE | Signal Str... |
|---|---|---|---|---|
| | | | | |



Input

Enter the Y Co-Ordinate

1500



RSU No:RSU7673

PortNo:5738 Latitude 35 Longitude 1500

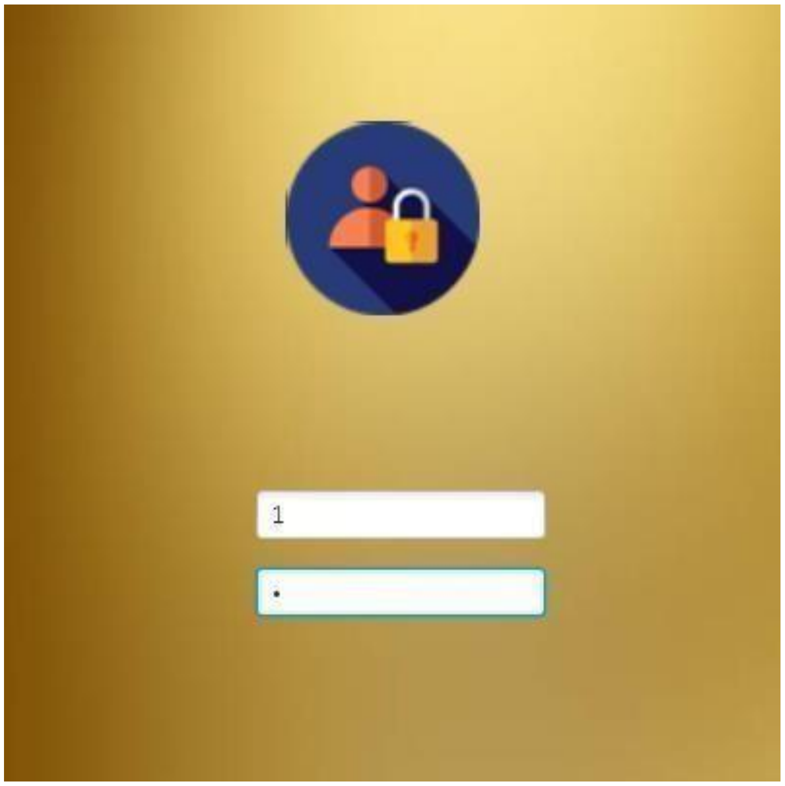
Neighbour

RSU6769

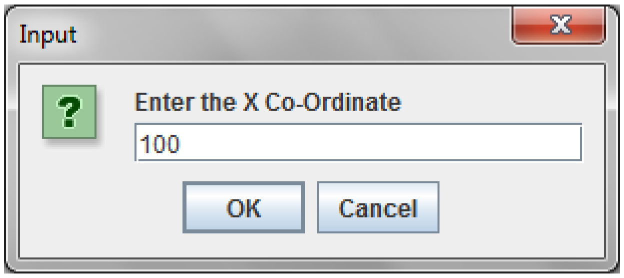
Receiver

| VEHI... | PORT... | LOCA... | RESPONSE MESSAGE | Signal Str... |
|---|---|---|---|---|
| | | | | |

LoginUser:


1
•

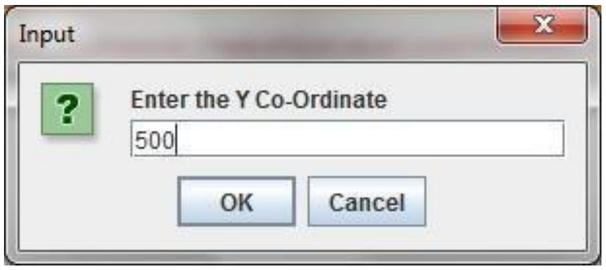
EnterX-CoordinateForVehicle



Enter the X Co-Ordinate

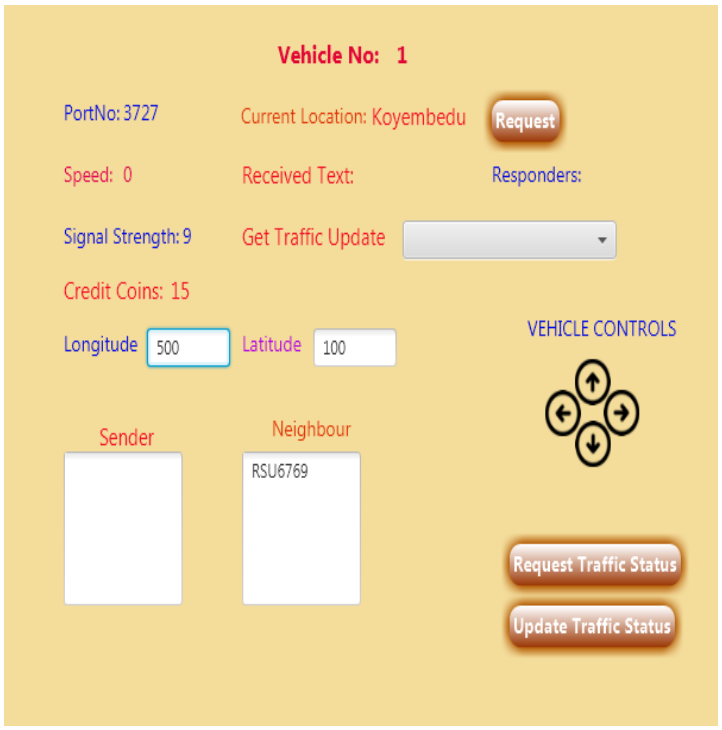
100

EnterY-CoordinateForVehicle



Enter the Y Co-Ordinate

500



Vehicle No: 1

PortNo: 3727

Current Location: Koyembedu

Speed: 0

Received Text:

Responders:

Signal Strength: 9

Get Traffic Update

Credit Coins: 15

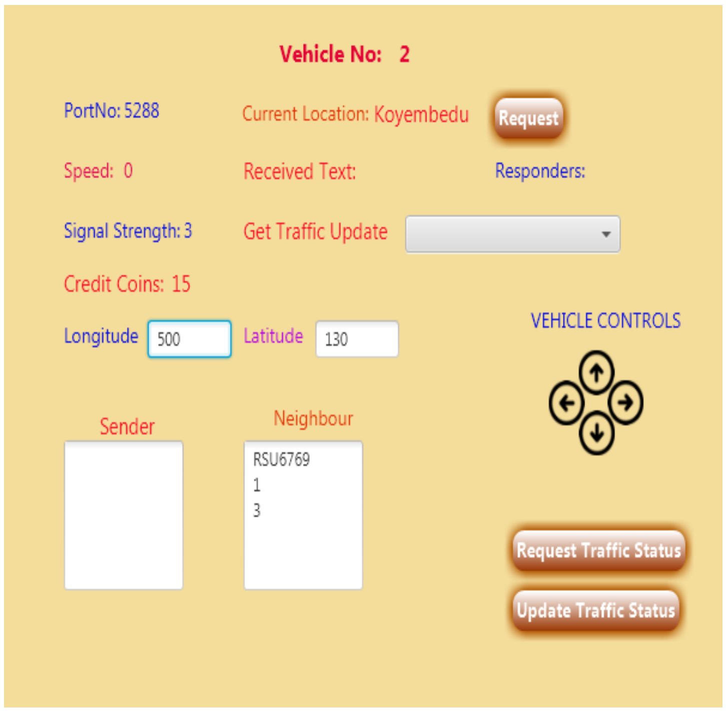
Longitude 500

Latitude 100

Sender

Neighbour

RSU6769



Vehicle No: 2

PortNo: 5288

Current Location: Koyembedu

Speed: 0

Received Text:

Responders:

Signal Strength: 3

Get Traffic Update

Credit Coins: 15

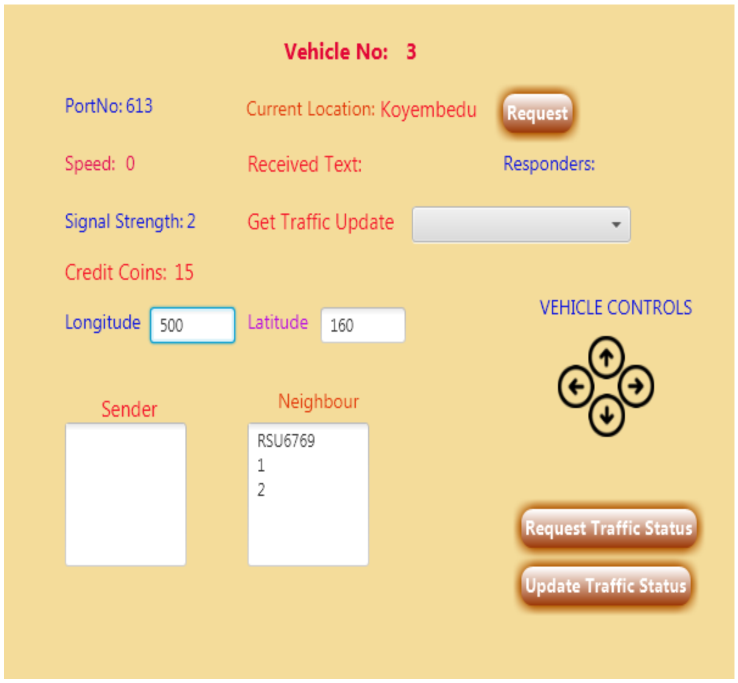
Longitude 500

Latitude 130

Sender

Neighbour

RSU6769
1
3


Vehicle No: 3

PortNo: 613

Current Location: Koyembedu

Speed: 0

Received Text:

Responders:

Signal Strength: 2

Get Traffic Update

Credit Coins: 15

Longitude 500

Latitude 160

Sender

Neighbour

RSU6769
1
2