

# 1.INTRODUCTION

Smart cities have drawn much attention due to the rapid growth of urbanization and the resulting pollution from traffic, in both academia and industry. Vehicular announcement networks 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. Additionally, it is also eco-friendly while decreasing the expenditure of many public resources by reducing the frequency of traffic jams and accidents. Blockchain is a novel decentralized ledger-based storage method. Satoshi firstly applied Blockchain into Bitcoin [3], which is a peer to peer e-cash system. Later, Blockchain gets more and more attention in e-commerce. Particularly, it has become a hot topic since Blockchain-based Bitcoin became popular. Moreover, in Blockchain-based networks, each node manages a copy of the whole or part of a database from the system. Thus, Blockchain-based networks are promising in recording credit data with the good properties of tamperresistance and decentralization, which is useful in VANETs. With the increasing privacy concerns of data [4]–[7], 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.

## 1.1 MOTIVATION

The motivation behind the project stems from recognizing the immense potential of vehicular announcement networks in enhancing communication within smart vehicles and smart transportation systems. However, two significant challenges hinder the effectiveness of such networks: the difficulty in forwarding announcements without compromising user identities and the lack of motivation among users to participate in forwarding announcements.

Addressing these challenges, the researchers propose CreditCoin, an innovative privacy-preserving incentive announcement network based on Blockchain technology. The aim is twofold: first, to enable anonymous and reliable forwarding of announcements, and second, to incentivize users to actively participate in sharing traffic information. CreditCoin tackles the anonymity issue by allowing non-deterministic signers to generate signatures and send announcements anonymously in environments that are not fully trusted. Additionally, by leveraging Blockchain technology, CreditCoin introduces incentives for users to share traffic information, ensuring tamper-resistant transactions and account information. Moreover, CreditCoin incorporates a trace manager feature to identify malicious users in anonymous announcements through related transactions, ensuring conditional privacy while motivating users to participate in forwarding announcements anonymously and reliably. The motivation behind CreditCoin lies in its potential to significantly enhance the efficiency and practicality of smart transportation systems by overcoming the identified challenges through a novel approach that prioritizes privacy, reliability, and user incentives. Experimental results further validate the effectiveness and feasibility of CreditCoin in simulations of smart transportation scenarios.

## **1.2 PROBLEM DEFINITION**

In the project discussed in the abstract, the researchers aim to tackle two primary challenges hindering the development of an effective vehicular announcement network: The first challenge lies in preserving user anonymity while forwarding announcements within the network. Ensuring anonymity is crucial for protecting the privacy of users in smart transportation systems. However, existing methods struggle to maintain anonymity effectively, potentially compromising user trust and impeding widespread adoption of the network. The second challenge involves motivating users to actively participate in forwarding announcements. Without adequate incentives, users may lack the motivation to share important traffic information, undermining the reliability and efficiency of the network in disseminating real-time updates within the transportation system.

These challenges underscore the critical need for a solution that can address privacy concerns while also incentivizing user involvement in forwarding announcements. By overcoming these obstacles, the researchers aim to enhance the efficiency and reliability of smart transportation systems.

### **1.3 OBJECTIVE OF THE PROJECT**

The objective of the project outlined in the abstract is to develop a novel vehicular announcement network named CreditCoin. The primary aim is to overcome two significant challenges prevalent in building effective vehicular announcement networks. Firstly, the project seeks to address the issue of anonymity preservation. This involves enabling the forwarding of announcements within the network while ensuring the anonymity of users. Preserving anonymity is crucial for protecting user privacy within smart transportation systems, where sensitive information is shared. Secondly, the project aims to tackle the problem of user motivation. By implementing incentive mechanisms, the goal is to encourage active participation from users in forwarding announcements. Without adequate incentives, users may lack the motivation to share vital traffic information, which could undermine the reliability and efficiency of the network. By addressing these challenges, the project ultimately aims to enhance the effectiveness and reliability of smart transportation systems through the development of CreditCoin.

## 2.LITERATURE SURVEY

[1] 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 auditability 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 auditability.

[2] Vehicular ad hoc networks (VANETs) have recently received significant attention in improving traffic safety and efficiency. However, communication trust and user privacy still present practical concerns to the deployment of VANETs, as many existing authentication protocols for VANETs either suffer from the heavy workload of downloading the latest revocation list from a remote authority or cannot allow drivers on the road to decide the trustworthiness of a message when the authentication on messages is anonymous. In this paper, to cope with these challenging concerns, we propose a new authentication protocol for VANETs in a decentralized group model by using a new group signature scheme. With the assistance of the new group signature scheme, the proposed authentication protocol is featured with threshold authentication, efficient revocation, unforgeability, anonymity, and traceability. In addition, the assisting group signature scheme may also be of independent interest, as it is characterized by efficient traceability and message linkability at the same time. Extensive analyses indicate that our proposed threshold anonymous authentication protocol is secure, and the verification of messages among vehicles can be accelerated by using batch message processing techniques.

[3] There has been significant interest and progress in the field of vehicular ad hoc networks over the last several years. VANETs comprise vehicle-to-vehicle and vehicle-to-infrastructure communications based on wireless local area network technologies. The distinctive set of candidate applications (e.g., collision warning and local traffic information for drivers), resources (licensed spectrum, rechargeable power source), and the environment (e.g., vehicular traffic flow patterns, privacy concerns) make the VANET a unique area of wireless communication. This

article gives an overview of the field, providing motivations, challenges, and a snapshot of proposed solutions.

[4] In the near future, most new vehicles will be equipped with short-range radios capable of communicating with other vehicles or with highway infrastructure at distances of at least one kilometer. The radios will allow new applications that will revolutionize the driving experience, providing everything from instant, localized traffic updates to warning signals when the car ahead abruptly brakes. While resembling traditional sensor and ad hoc networks in some respects, vehicular networks pose a number of unique challenges. For example, the information conveyed over a vehicular network may affect life-or-death decisions, making fail-safe security a necessity. However, providing strong security in vehicular networks raises important privacy concerns that must also be considered. To address these challenges, we propose a set of security primitives that can be used as the building blocks of secure applications. The deployment of vehicular networks is rapidly approaching, and their success and safety will depend on viable security solutions acceptable to consumers, manufacturers and governments.

[5] 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.

### **3. ANALYSIS**

#### **3.1 EXISTING SYSTEM:**

Vehicle announcement can help people avoid traffic jams and give them advance notice of any accident or event that may occur along a route before they choose that route, but it has some drawbacks, including user privacy leakage because their current location and vehicle ID will be made public and they could use this information to kidnap people, as well as a lack of user motivation to participate in the announcement.

##### **3.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

1. User privacy leakage
2. Lack of user motivation to participate in the announcement

#### **3.2 Proposed System:**

In this paper introduces a concept called Blockchain and incentive-based privacy-preserving announcement in vehicular networks to address the aforementioned issue. Here we adds another concept to avoid incentive lock where user incentives will expire at the end of the day so they may use this incentive to post their new data to new users. The proposed paper encourages users to announce their location, event, or others received data by providing INCENTIVES and to earn incentive all peoples will get involved in announcement.

##### **3.2.1 Advantages of proposed system:**

1. User privacy leakage will be avoided
2. Provide security to data

### **3.3 SYSTEM REQUIREMENT SPECIFICATION**

#### **SOFTWARE REQUIREMENTS**

1. Visual Studio Community Version
2. Nodejs ( Version 12.3.1)

3. Python IDEL ( Python 3.7 )

## **HARDWARE REQUIREMENTS**

1. Operating System : Windows Only

2. Processor : i5 and above

3. Ram : 4gb and above

4. Hard Disk : 50 GB

### **3.3.1 PURPOSE**

The purpose of the project outlined in the abstract is to devise a solution to the challenges facing vehicular announcement networks, particularly within smart transportation systems. By introducing CreditCoin, the researchers aim to create an innovative network that addresses two critical issues: maintaining user anonymity while forwarding announcements and incentivizing user participation in sharing traffic information. The overarching goal is to enhance the efficiency and reliability of smart transportation systems by providing a privacy-preserving and incentive-driven framework for communication among vehicles. Through CreditCoin, the project seeks to establish a robust infrastructure that fosters trust, encourages active engagement from users, and facilitates the seamless exchange of traffic data while safeguarding user privacy and ensuring data integrity.

### **3.3.2 SCOPE**

The scope of the project outlined in the abstract encompasses several key aspects essential for the development and implementation of the CreditCoin vehicular announcement network. Firstly, the project aims to explore and address the challenges associated with maintaining user anonymity while forwarding announcements within smart transportation systems. This involves devising mechanisms that allow for the dissemination of traffic information without compromising the privacy of individual users. Secondly, the project seeks to devise incentive mechanisms to motivate users to actively participate in sharing traffic data. By offering rewards or benefits, the aim is to encourage widespread engagement and collaboration among users, thereby enhancing the reliability and effectiveness of the network. Additionally, the scope includes implementing privacy-preserving measures to safeguard sensitive user information and ensure the tamper-resistance of transactions and account data. Leveraging Blockchain technology, the project aims

to provide a secure and trustworthy framework for communication within the network. Furthermore, the project encompasses the development of traceability mechanisms to identify and deter malicious users, thereby maintaining the integrity of the network. Overall, the scope of the project spans the design, implementation, and evaluation of CreditCoin as a comprehensive solution for enhancing communication and data sharing within smart transportation systems while prioritizing privacy, reliability, and user engagement.

### **3.3.3 OVERALL DESCRIPTION**

The project outlined in the abstract introduces CreditCoin, a novel vehicular announcement network designed to address key challenges within smart transportation systems. CreditCoin aims to revolutionize communication among vehicles by tackling two primary issues: preserving user anonymity while forwarding announcements and incentivizing user participation in sharing traffic information. To achieve these objectives, the project proposes innovative solutions, including non-deterministic signature generation for anonymous announcements and incentive mechanisms to motivate user engagement. Leveraging Blockchain technology, CreditCoin ensures the privacy, security, and integrity of transactions and account data, while traceability features help identify and deter malicious users. By providing a comprehensive framework that prioritizes privacy, reliability, and user incentives, CreditCoin seeks to enhance the efficiency and effectiveness of smart transportation systems, fostering trust, collaboration, and seamless data exchange among vehicles. Through extensive experimentation and evaluation, the project aims to demonstrate the practicality and feasibility of CreditCoin in real-world transportation scenarios, paving the way for its widespread adoption and integration into future smart mobility solutions



## 4. DESIGN

### 4.1 SYSTEM ARCHITECTURE:

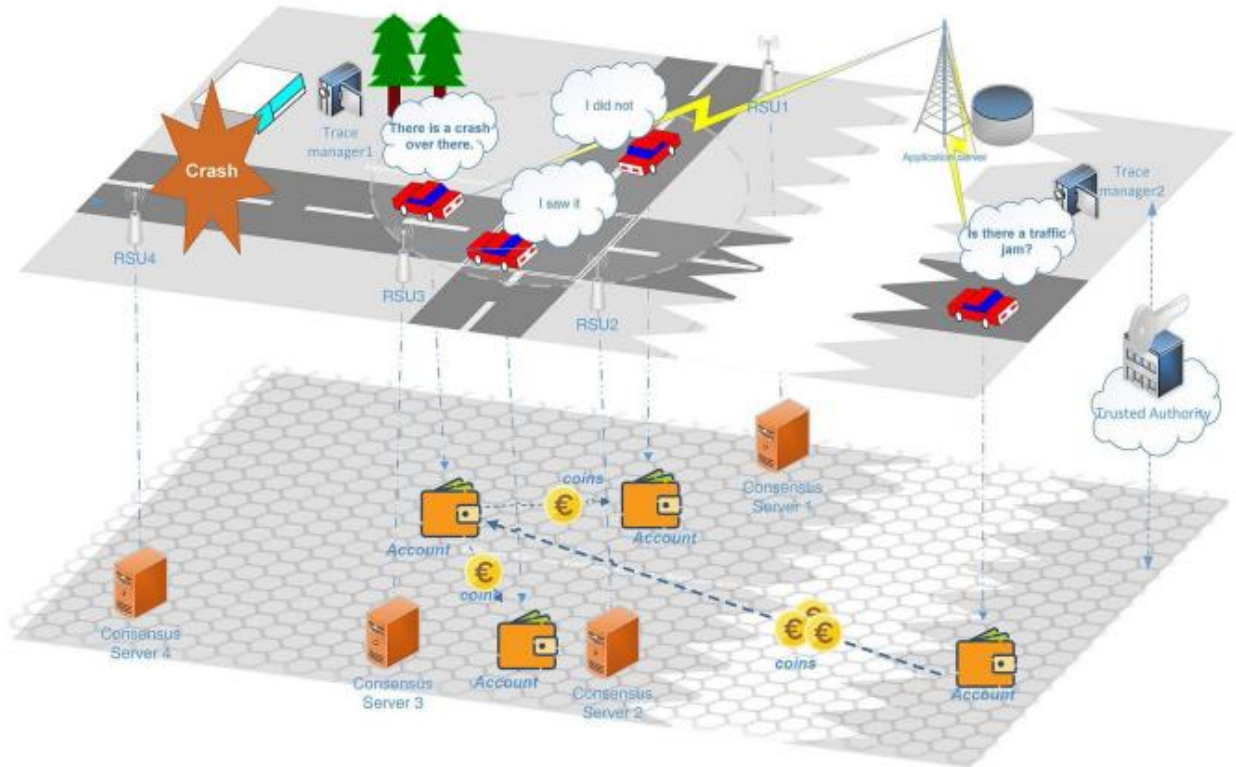


Fig. 4.1 System Architecture

### 4.2 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

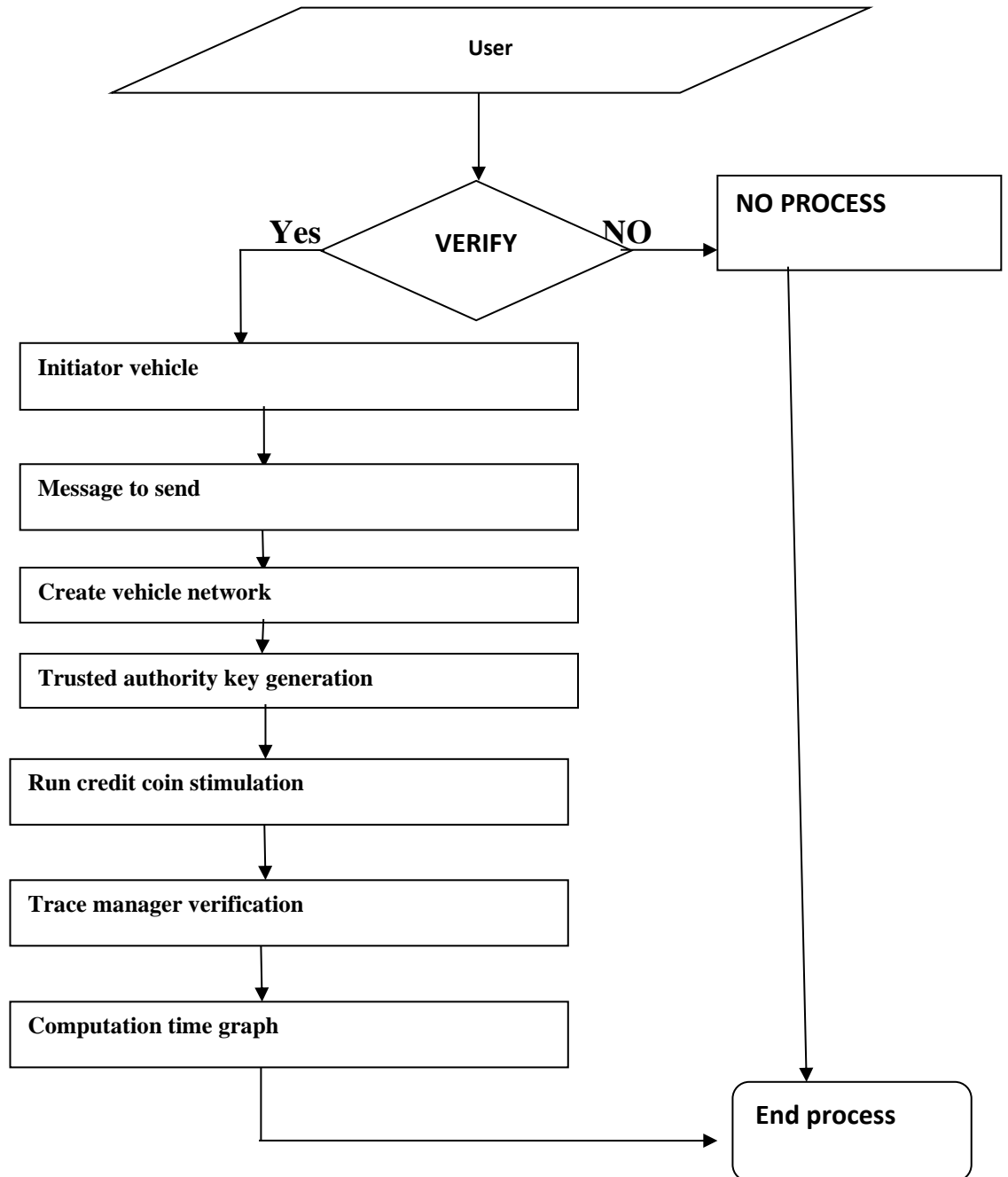


Fig.4.2 Data Flow Diagram

### **4.3 UML DIAGRAMS:**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

### **4.4 Use case diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as

use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

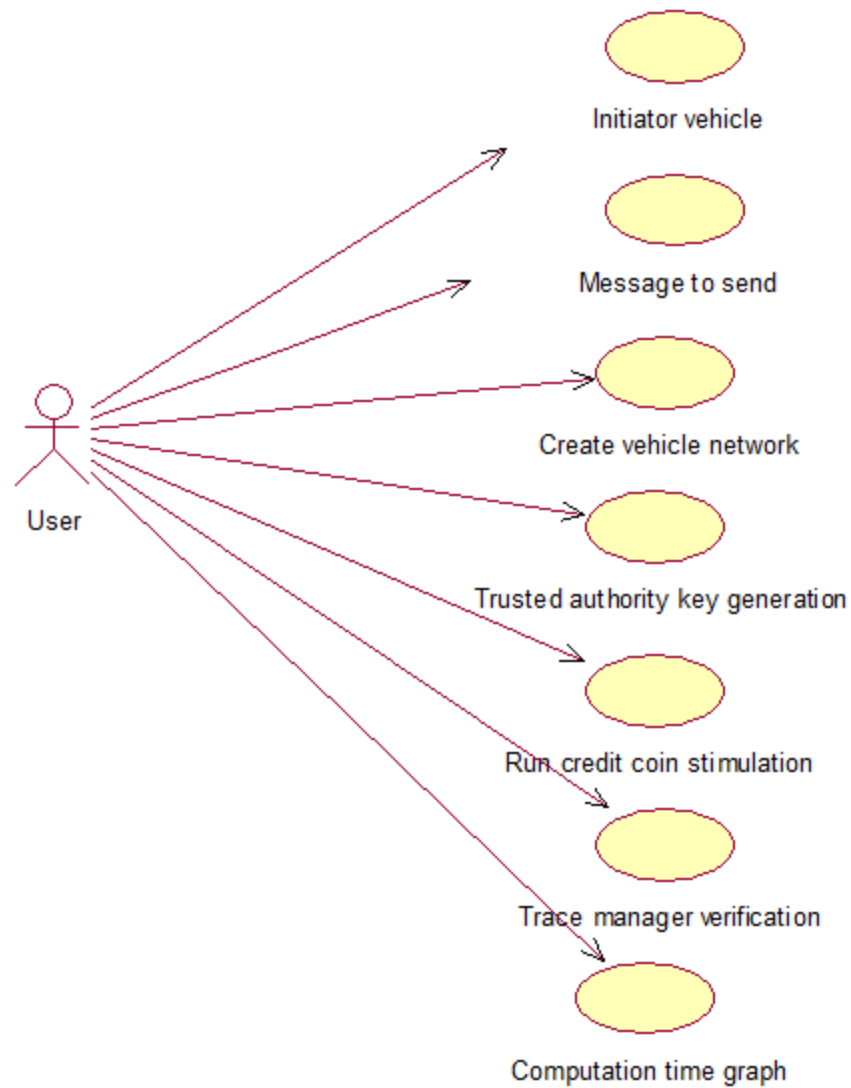


Fig 4.4.Use Case Diagram

## 4.5 Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

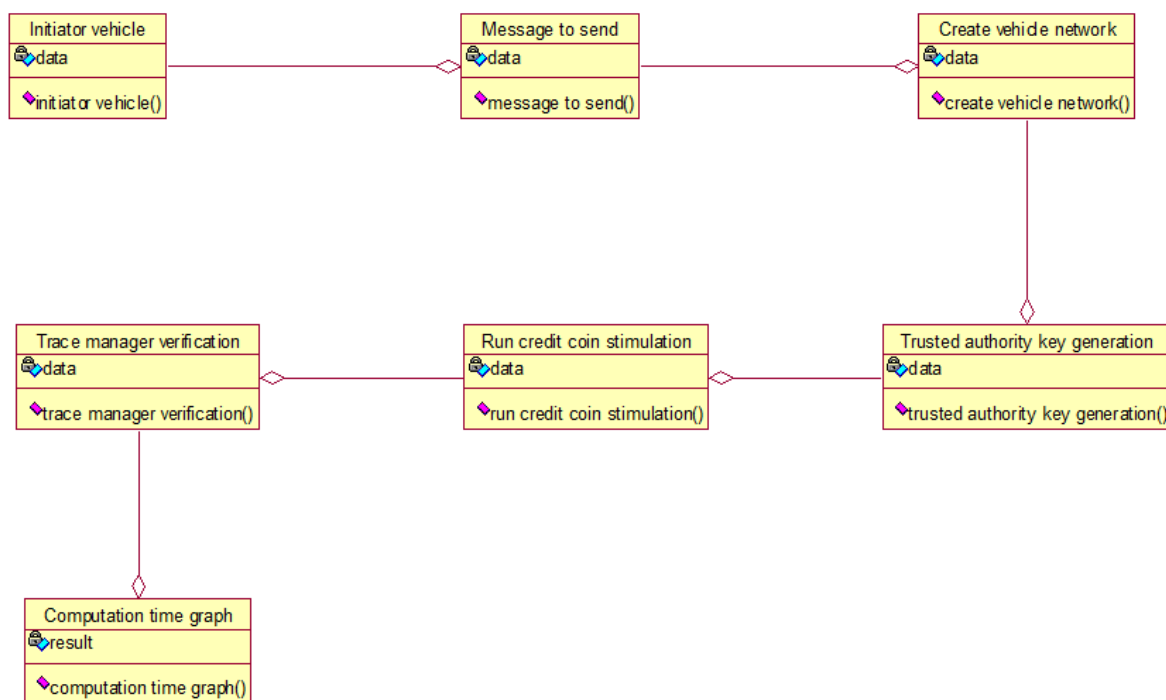


Fig.4.5 Class Diagram

## 4.6 Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

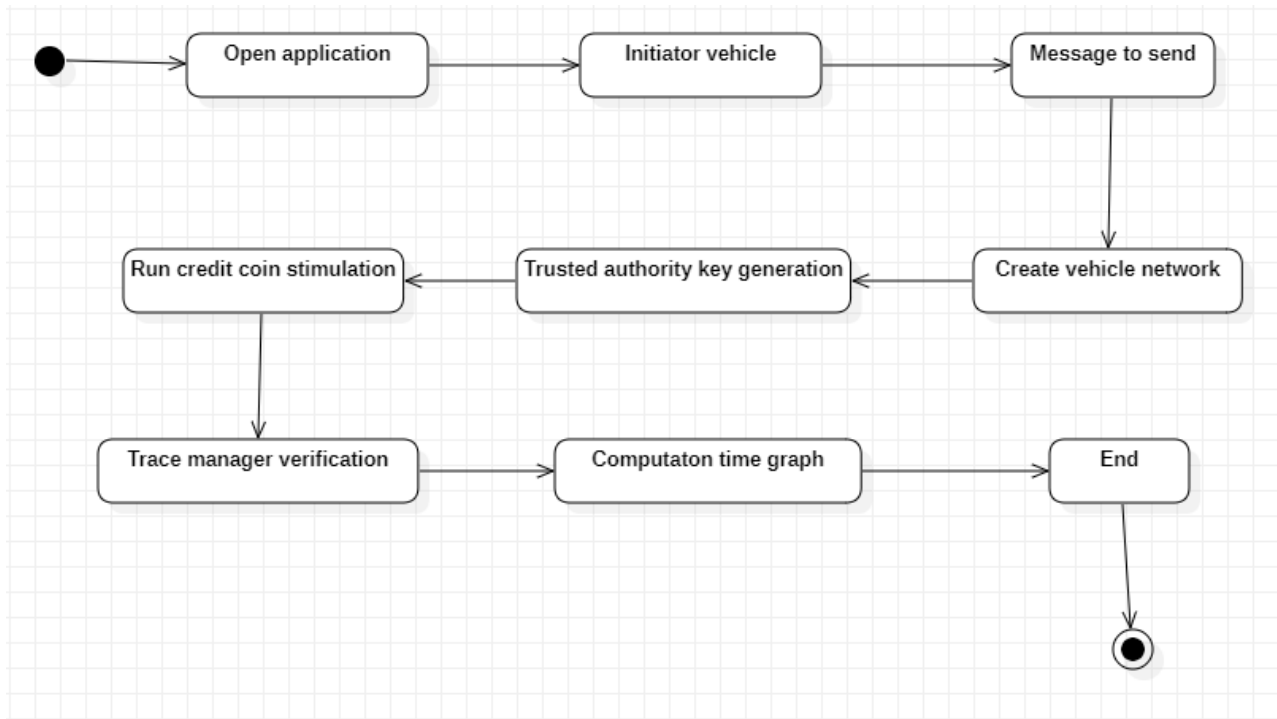


Fig.4.6 Activity Diagram

#### 4.7 Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

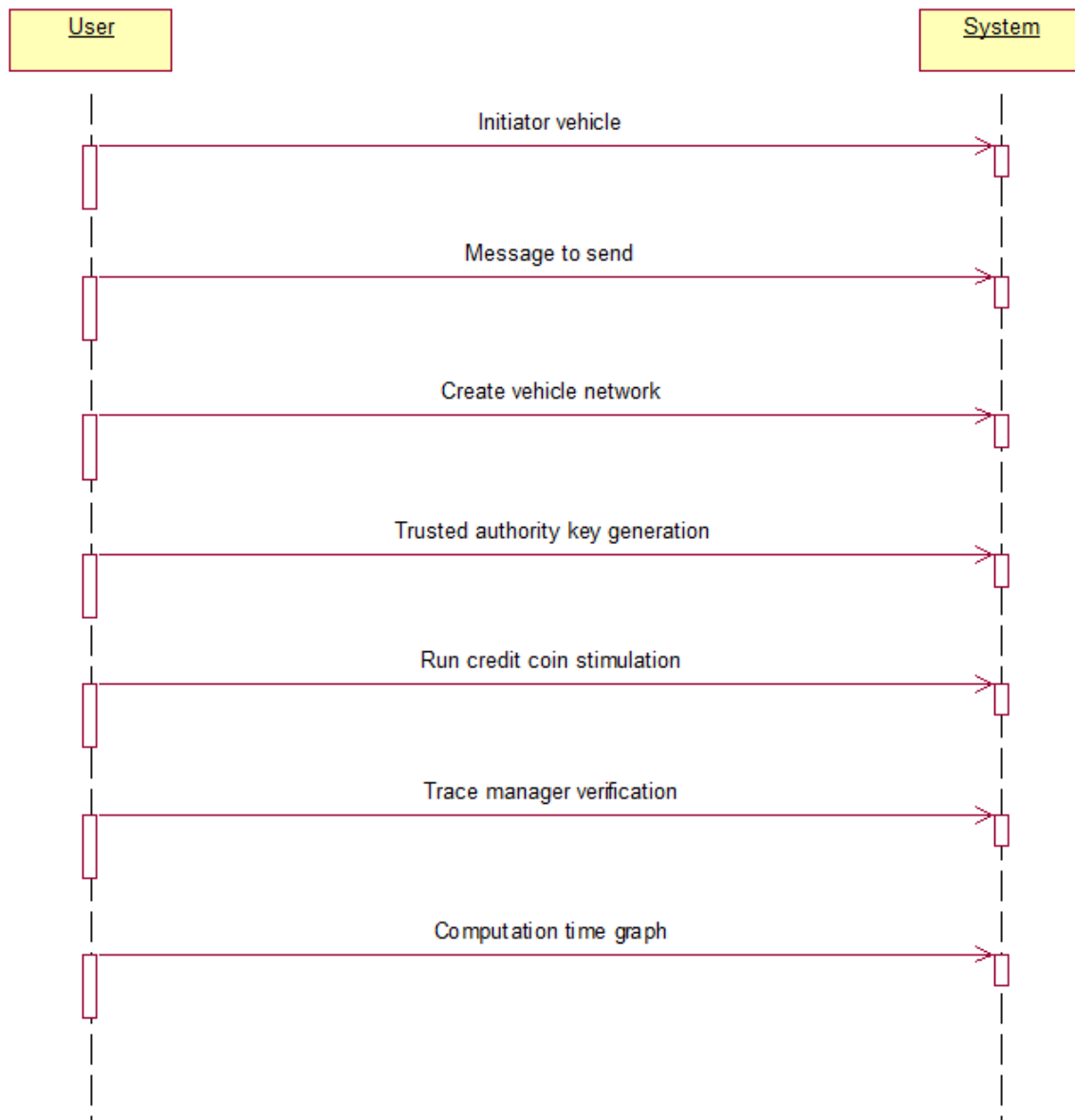


fig.4.7 Sequence Diagram

#### 4.8 Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions.

The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

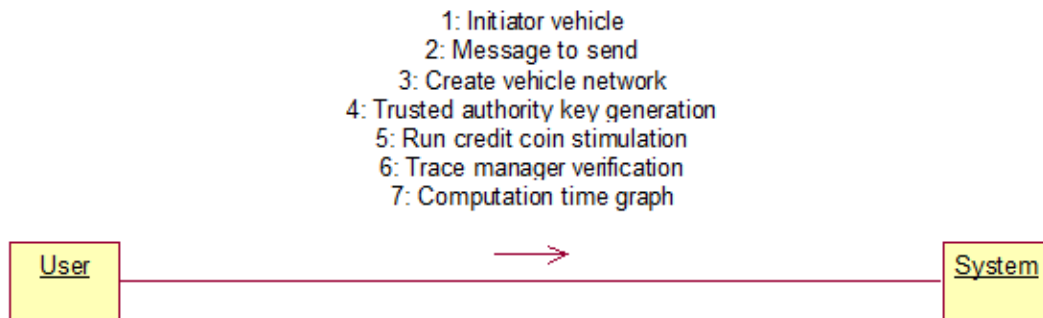


Fig.4.8 Collaboration Diagram

#### 4.9 Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



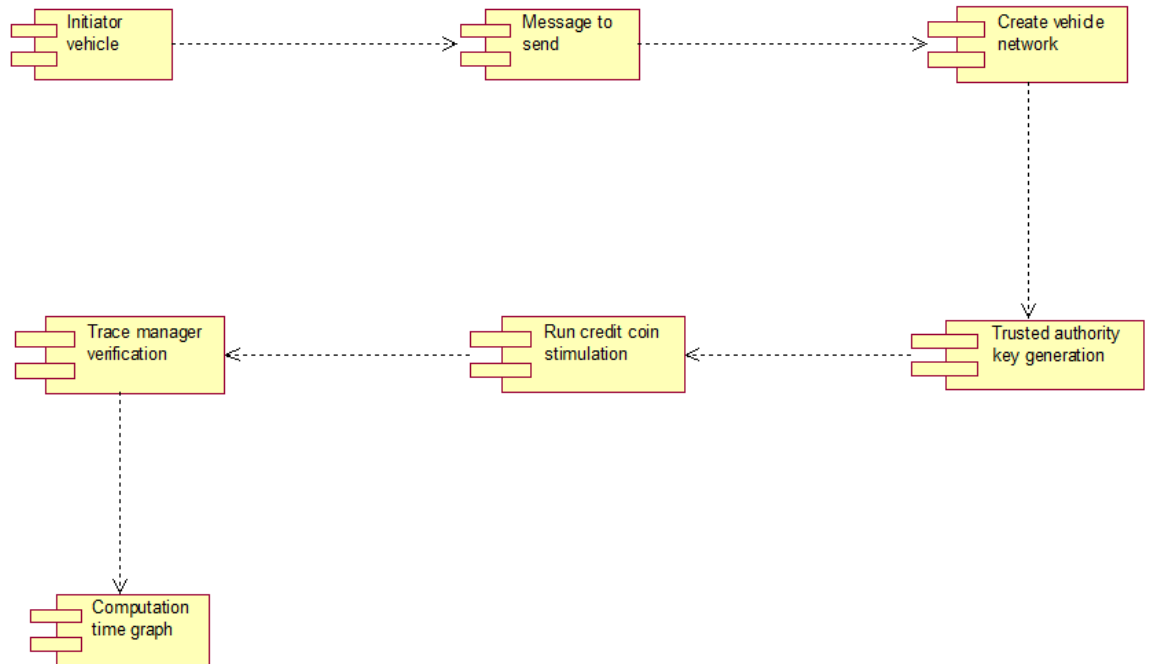


Fig.4.9 Component Diagram

#### 4.10 Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

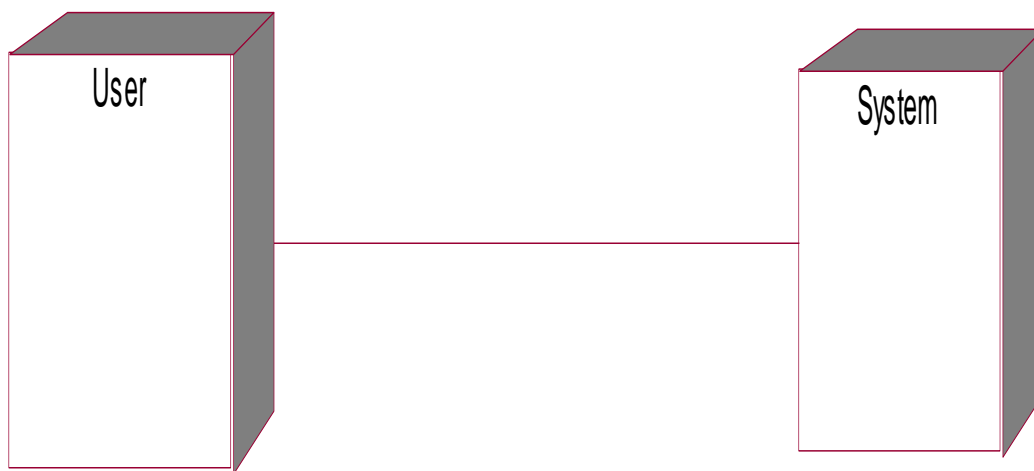


Fig.4.10 Deployment Diagram

## 5. IMPLEMENTATION

Vehicular announcement can help to avoid traffic jams and allow peoples to know any accident or event occur at any route prior of choosing that route but this announcement contains some flaws as user privacy leakage who involved in announcement as their current location and vehicle ID will be exposed to other peoples and they can misuse this location information to kidnap peoples and other flaw is lack of motivation in user to involved themselves to announce that information to other vehicles which are in their range.

To overcome from above problem author of this paper has introduce concept called Blockchain & incentive based privacy preserving announcement in vehicular network. In propose paper is motivating users to announce their location and event or others received data by providing INCENTIVES and to earn incentive all peoples will get involved in announcement and author adding another concept to avoid incentive lock where user incentives will be expired at the end of the day so they may use this incentive to post their new data to new users.

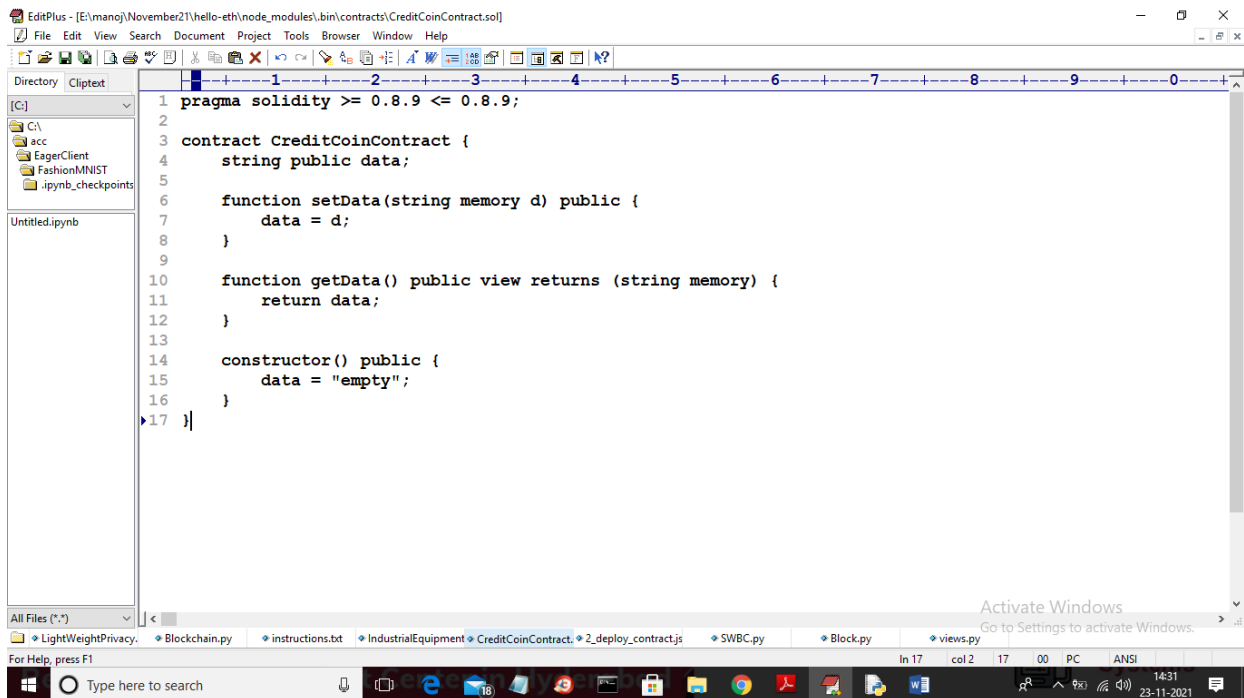
To provide privacy to user author is using symmetric ring signature where all users like INITIATOR who send request and the REPLIERS who reply to initiator requestor will signed to one public, private and secret KEY and then all their ID's and data will be encrypted using public key and the receiver can decrypt data using their private key. To prevent attacks author using Blockchain based verification which will store data as Block of chain and each block will be associated with one unique hashcode and before adding any new transaction or Block then Blockchain verify all previous blocks hashcode and if all blocks verification successful then data consider as intact or secured and if not successful then data will be consider as attacked and Blockchain will not allow any block of data to get tamper so Blockchain will provide security to data and by using symmetric encryption user privacy leakage will be avoided.

### 5.1 modules

- 1) Role settings: Here we have various users such as Initiator (vehicle which send request), Replier (vehicle which reply to initiator), Trusted authority (responsible for key management), Verifier (Trace Manager which perform verification)

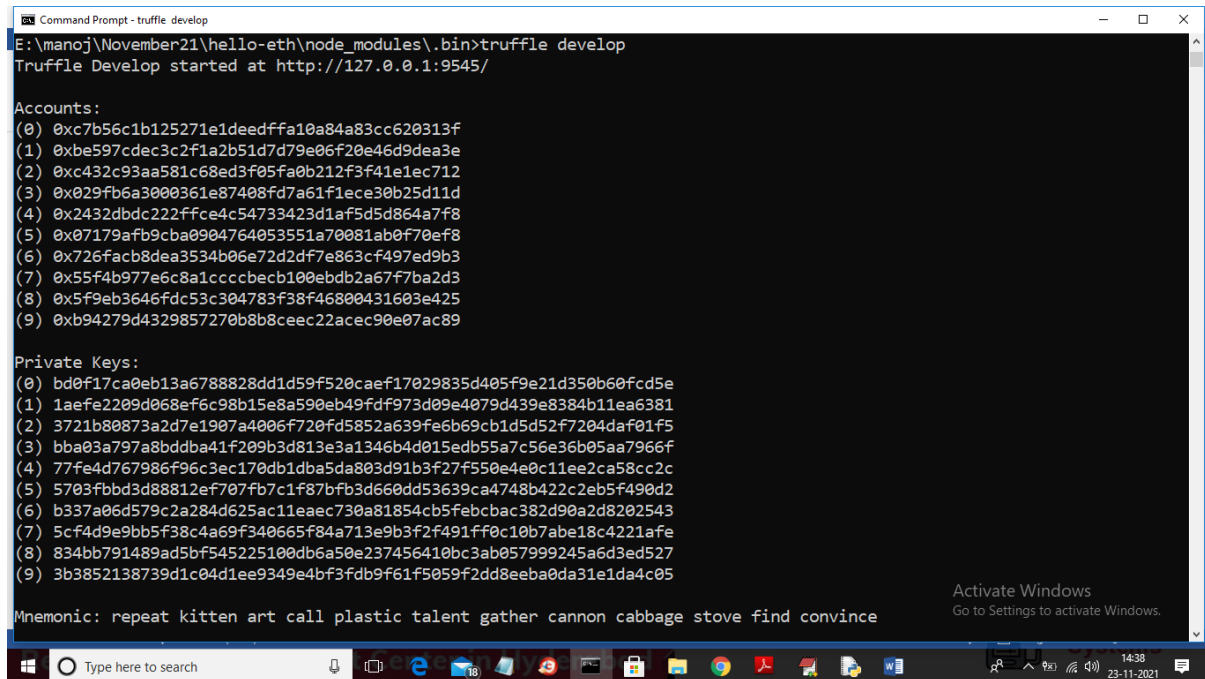
- 2) Packets settings: based on role Echo Announcement packets will be generated as request or reply
- 3) Request Packet: sent by initiator to get reply from replier so they must agree on announcement
- 4) Reply Packet: sent by replier willing to join announcement with requester
- 5) Announcement Aggregate Packet (AGP): packet sent from requester to verifier from verification
- 6) Generate Key Trusted Authority: generate and manage key by Trusted Authority
- 7) Run CreditCoin Simulation: using this module initiator vehicle will send request to nearer repliers and then increment incentives of involving replier. Each request and reply packet will be encrypted to avoid privacy leakage
- 8) Trace Manager Verification: using this module verifier will read all blocks or packets and then verify their hashcode
- 9) Computation Time Graph: using this module we will plot request, reply and key generation computation time

All the packets will be saved in Blockchain using SOLIDITY Contract below code



```
1 pragma solidity >= 0.8.9 <= 0.8.9;
2
3 contract CreditCoinContract {
4     string public data;
5
6     function setData(string memory d) public {
7         data = d;
8     }
9
10    function getData() public view returns (string memory) {
11        return data;
12    }
13
14    constructor() public {
15        data = "empty";
16    }
17 }
```

Above solidity function will be deployed and called by Blockchain to store vehicle information. Below are the instructions to deploy and call contract from Blockchain. Now we need to deploy above code to store and access CreditCoin data into Ethereum Tool and to deploy go inside 'hello-eth\node\_modules\.bin' folder and in this double click on 'runBlockchain.bat' file to get below screen



```
Command Prompt - truffle develop
E:\manoj\November21\hello-eth\node_modules\.bin>truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0xc7b56c1b125271e1deedffa10a84a83cc620313f
(1) 0xbe597cdec3c2f1a2b51d7d79e06f20e46d9dea3e
(2) 0xc432c93aa581c68ed3f05fa0b212f3f41e1ec712
(3) 0x029fb6a3000361e87408fd7a61f1ece30b25d11d
(4) 0x2432dbdc222ffce4c54733423d1af5d5d864a7f8
(5) 0x07179afb9cba0904764053551a70081ab0f70ef8
(6) 0x726facb8dea3534b06e72d2df7e863cf497ed9b3
(7) 0x55f4b977e6c8a1ccccbecb100ebdb2a67f7ba2d3
(8) 0x5f9eb3646fcd53c304783f38f46800431603e425
(9) 0xb94279d4329857270b8b8ceec22acac90e07ac89

Private Keys:
(0) bd0f17ca0eb13a6788828dd1d59f520caef17029835d405f9e21d350b60fcd5e
(1) 1aefe2209d068ef6c98b15e8a590eb49fd973d09e4079d439e8384b11ea6381
(2) 3721b80873a2d7e1907a4006f720fd5852a639fe6b69cb1d5d52f7204daf01f5
(3) bba03a797a8bddba41f209b3d813e3a1346b4d015edb55a7c56e36b05aa7966f
(4) 77fe4d767986f96c3ec170db1dba5da803d91b3f27f550e4e0c11ee2ca58cc2c
(5) 5703fbbd3d88812ef707fb7c1f87bfb3d660dd53639ca4748b422c2eb5f490d2
(6) b337a06d579c2a284d625ac11eac730a81854cb5febcbac382d90a2d8202543
(7) 5cf4d9e9bb5f38c4a69f340665f84a713e9b3f2f491ff0c10b7abe18c4221afe
(8) 834bb791489ad5bf545225100db6a50e237456410bc3ab057999245a6d3ed527
(9) 3b3852138739d1c04d1ee9349e4bf3fdb9f61f5059f2dd8eeba0da31e1da4c05

Mnemonic: repeat kitten art call plastic talent gather cannon cabbage stove find convince

Activate Windows
Go to Settings to activate Windows.
```

In above screen we can see Blockchain accounts and private keys are generated and now to deploy code type 'migrate' and press enter key to deploy code and to get contract access ADDRESS like below screen

```
Command Prompt - truffle develop

(2) 0xc432c93aa581c68ed3f05fa0b212f3f41e1ec712
(3) 0x029fb6a3000361e87408fd7a61f1ece30b25d11d
(4) 0x2432dbdc222ffce4c54733423d1af5d5d864a7f8
(5) 0x07179afb9c9a0904764053551a70081ab0f70ef8
(6) 0x726facb8dea3534b06e72d2df7e863cf497ed9b3
(7) 0x55f4b977e6c8a1ccccbecb10ebdb2a67f7ba2d3
(8) 0x5f9eb3646fdc53c304783f38f46800431603e425
(9) 0xb94279d4329857270b8b8ceec22acac90e07ac89

Private Keys:
(0) bd0f17ca0eb13a6788828dd1d59f520caef17029835d405f9e21d350b60fcd5e
(1) 1aefe2209d068ef6c98b15e8a590eb49fd973d09e4079d439e8384b11ea6381
(2) 3721b80873a2d7e1907a4006f720fd5852a639fe6b69cb1d5d52f7204daf01f5
(3) bba03a797a8bddba41f209b3d813e3a1346b4d015edb55a7c56e36b05aa7966f
(4) 77fe4d767986f96c3ec170db1dba5da803d91b3f27f550e4e0c11ee2ca58cc2c
(5) 5703fbbd3d88812ef707fb7c1f87bfb3d660dd53639ca4748b422c2eb5f490d2
(6) b337a06d579c2a284d625ac11eae730a81854cb5febcbac382d90a2d8202543
(7) 5cf4d9e9bb5f38c4a69f340665f84a713e9b3f2f491ff0c10b7abe18c4221afe
(8) 834bb791489ad5bf545225100db6a50e237456410bc3ab057999245a6d3ed527
(9) 3b3852138739d1c04d1ee9349e4bf3fdb9f61f5059f2dd8eeba0da31e1da4c05

Mnemonic: repeat kitten art call plastic talent gather cannon cabbage stove find convince

Important : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)> migrate
```

After running above command will get below screen

```
Command Prompt - truffle develop

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:          0.000497684 ETH

2_deploy_contract.js
=====

Deploying 'CreditCoinContract'
-----
> transaction hash:    0xc594b40a0f15fd21d1918e802f35e9d4749371094ec8041542ed7d27329c8341
> Blocks: 0           Seconds: 0
> contract address:   0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058
> block number:       3
> block timestamp:    1637644285
> account:            0xc7B56c1B125271E1dEeDffA10a84a83cC620313f
> balance:            99.998666926
> gas used:           375182 (0x5b98e)
> gas price:          2 gwei
> value sent:         0 ETH
> total cost:         0.000750364 ETH

> Saving migration to chain.
> Saving artifacts
-----
```

In above screen we can see Credit Coin contract deployed to Blockchain and we can call above code to store all vehicle communication details. In below code will specify above contract address to access Blockchain

```

*CreditCoin.py - E:\manoj\November21\CreditCoin\CreditCoin.py (3.7.0)
File Edit Format Run Options Window Help

global vehicle_list
global key
global ta
global compute_time

def saveDataBlockchain(currentData): #calling to save data in blockchain
    global details
    global contract
    blockchain_address = 'http://127.0.0.1:9545'
    web3 = Web3(HTTPProvider(blockchain_address))
    web3.eth.defaultAccount = web3.eth.accounts[0]
    compiled_contract_path = 'CreditCoinContract.json'
    deployed_contract_address = '0x1D04fb45C10dc9C3f32cbaA60464c8107D4D4058'
    with open(compiled_contract_path) as file:
        contract_json = json.load(file) # load contract info as JSON
        contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its functions
    file.close()
    contract = web3.eth.contract(address=deployed_contract_address, abi=contract_abi)
    readDetails()
    details=currentData
    msg = contract.functions.setData(details).transact()
    tx_receipt = web3.eth.waitForTransactionReceipt(msg)

def readDetails(): #calling to read data from blockchain
    global details
    blockchain_address = 'http://127.0.0.1:9545' #Blockchain connection IP
    web3 = Web3(HTTPProvider(blockchain_address))
    web3.eth.defaultAccount = web3.eth.accounts[0]
    compiled_contract_path = 'CreditCoinContract.json' #industrial contract code
    deployed_contract_address = '0x1D04fb45C10dc9C3f32cbaA60464c8107D4D4058' #hash address to access industrail contract
    with open(compiled_contract_path) as file:
        contract_json = json.load(file) # load contract info as JSON
        contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its functions
    file.close()
    contract = web3.eth.contract(address=deployed_contract_address, abi=contract_abi) #now calling contract to access data
    details = contract.functions.getData().call()
    if len(details) > 0:
        if 'empty' in details:
            details = details[5:len(details)]
    return details

```

In above screen you can read red colour comments to know how Blockchain is calling to store and read data.

## 5.2 SAMPLE CODE:

```

import os
import sys
import signal
import atexit
from hashlib import sha256
import json
import time

from flask import Flask, request
import requests

```

**class Block:**

```

    def __init__(self, index, transactions, timestamp, previous_hash, nonce=0):
        self.index = index
        self.transactions = transactions
        self.timestamp = timestamp
        self.previous_hash = previous_hash
        self.nonce = nonce

```

```

def compute_hash(self):
    """
    A function that return the hash of the block contents.
    """
    block_string = json.dumps(self.__dict__, sort_keys=True)
    return sha256(block_string.encode()).hexdigest()

```

```

class Blockchain:

```

```

    # difficulty of our PoW algorithm
    difficulty = 2

```

```

    def __init__(self, chain=None):
        self.unconfirmed_transactions = []
        self.chain = chain
        if self.chain is None:
            self.chain = []
            self.create_genesis_block()

```

```

    def create_genesis_block(self):
        """
        A function to generate genesis block and appends it to
        the chain. The block has index 0, previous_hash as 0, and
        a valid hash.
        """
        genesis_block = Block(0, [], 0, "0")
        genesis_block.hash = genesis_block.compute_hash()
        self.chain.append(genesis_block)

```

```

    @property
    def last_block(self):
        return self.chain[-1]

```

```

    def add_block(self, block, proof):
        """
        A function that adds the block to the chain after verification.
        Verification includes:
        * Checking if the proof is valid.
        * The previous_hash referred in the block and the hash of latest block
          in the chain match.
        """
        previous_hash = self.last_block.hash

        if previous_hash != block.previous_hash:
            raise ValueError("Previous hash incorrect")

```

```

    if not Blockchain.is_valid_proof(block, proof):
        raise ValueError("Block proof invalid")

    block.hash = proof
    self.chain.append(block)

    @staticmethod
    def proof_of_work(block):
        """
        Function that tries different values of nonce to get a hash
        that satisfies our difficulty criteria.
        """
        block.nonce = 0

        computed_hash = block.compute_hash()
        while not computed_hash.startswith('0' * Blockchain.difficulty):
            block.nonce += 1
            computed_hash = block.compute_hash()

        return computed_hash

    def add_new_transaction(self, transaction):
        self.unconfirmed_transactions.append(transaction)

    @classmethod
    def is_valid_proof(cls, block, block_hash):
        """
        Check if block_hash is valid hash of block and satisfies
        the difficulty criteria.
        """
        return (block_hash.startswith('0' * Blockchain.difficulty) and
                block_hash == block.compute_hash())

    @classmethod
    def check_chain_validity(cls, chain):
        result = True
        previous_hash = "0"

        for block in chain:
            block_hash = block.hash
            # remove the hash field to recompute the hash again
            # using `compute_hash` method.
            delattr(block, "hash")

            if not cls.is_valid_proof(block, block_hash) or \
               previous_hash != block.previous_hash:

```



```

        result = False
        break

    block.hash, previous_hash = block_hash, block_hash

    return result

def mine(self):
    """
    This function serves as an interface to add the pending
    transactions to the blockchain by adding them to the block
    and figuring out Proof Of Work.
    """
    if not self.unconfirmed_transactions:
        return False

    last_block = self.last_block

    new_block = Block(index=last_block.index + 1,
                       transactions=self.unconfirmed_transactions,
                       timestamp=time.time(),
                       previous_hash=last_block.hash)

    proof = self.proof_of_work(new_block)
    self.add_block(new_block, proof)

    self.unconfirmed_transactions = []

    return True

app = Flask(__name__)

# the node's copy of blockchain
blockchain = None

# the address to other participating members of the network
peers = set()

# endpoint to submit a new transaction. This will be used by
# our application to add new data (posts) to the blockchain
@app.route('/new_transaction', methods=['POST'])
def new_transaction():
    tx_data = request.get_json()
    required_fields = ["author", "content"]

```

```

    for field in required_fields:
        if not tx_data.get(field):
            return "Invalid transaction data", 404

    tx_data["timestamp"] = time.time()

    blockchain.add_new_transaction(tx_data)

    return "Success", 201

chain_file_name = os.environ.get('DATA_FILE')

def create_chain_from_dump(chain_dump):
    generated_blockchain = Blockchain()
    for idx, block_data in enumerate(chain_dump):
        if idx == 0:
            continue # skip genesis block
        block = Block(block_data["index"],
                      block_data["transactions"],
                      block_data["timestamp"],
                      block_data["previous_hash"],
                      block_data["nonce"])
        proof = block_data['hash']
        generated_blockchain.add_block(block, proof)
    return generated_blockchain

# endpoint to return the node's copy of the chain.
# Our application will be using this endpoint to query
# all the posts to display.
@app.route('/chain', methods=['GET'])
def get_chain():
    chain_data = []
    for block in blockchain.chain:
        chain_data.append(block.__dict__)
    return json.dumps({"length": len(chain_data),
                      "chain": chain_data,
                      "peers": list(peers)})

def save_chain():
    if chain_file_name is not None:
        with open(chain_file_name, 'w') as chain_file:

```

```

    chain_file.write(get_chain())

def exit_from_signal(signum, stack_frame):
    sys.exit(0)

atexit.register(save_chain)
signal.signal(signal.SIGTERM, exit_from_signal)
signal.signal(signal.SIGINT, exit_from_signal)

if chain_file_name is None:
    data = None
else:
    with open(chain_file_name, 'r') as chain_file:
        raw_data = chain_file.read()
        if raw_data is None or len(raw_data) == 0:
            data = None
        else:
            data = json.loads(raw_data)

if data is None:
    # the node's copy of blockchain
    blockchain = Blockchain()
else:
    blockchain = create_chain_from_dump(data['chain'])
    peers.update(data['peers'])

# endpoint to request the node to mine the unconfirmed
# transactions (if any). We'll be using it to initiate
# a command to mine from our application itself.
@app.route('/mine', methods=['GET'])
def mine_unconfirmed_transactions():
    result = blockchain.mine()
    if not result:
        return "No transactions to mine"
    else:
        # Making sure we have the longest chain before announcing to the network
        chain_length = len(blockchain.chain)
        consensus()
        if chain_length == len(blockchain.chain):
            # announce the recently mined block to the network
            announce_new_block(blockchain.last_block)
        return "Block #{0} is mined.".format(blockchain.last_block.index)

```

```

# endpoint to add new peers to the network.
@app.route('/register_node', methods=['POST'])
def register_new_peers():
    node_address = request.get_json()["node_address"]
    if not node_address:
        return "Invalid data", 400

    # Add the node to the peer list
    peers.add(node_address)

    # Return the consensus blockchain to the newly registered node
    # so that he can sync
    return get_chain()

@app.route('/register_with', methods=['POST'])
def register_with_existing_node():
    """
    Internally calls the `register_node` endpoint to
    register current node with the node specified in the
    request, and sync the blockchain as well as peer data.
    """
    node_address = request.get_json()["node_address"]
    if not node_address:
        return "Invalid data", 400

    data = {"node_address": request.host_url}
    headers = {'Content-Type': 'application/json'}

    # Make a request to register with remote node and obtain information
    response = requests.post(node_address + "/register_node",
                             data=json.dumps(data), headers=headers)

    if response.status_code == 200:
        global blockchain
        global peers
        # update chain and the peers
        chain_dump = response.json()['chain']
        blockchain = create_chain_from_dump(chain_dump)
        peers.update(response.json()['peers'])
        return "Registration successful", 200
    else:
        # if something goes wrong, pass it on to the API response
        return response.content, response.status_code

```

```

# endpoint to add a block mined by someone else to
# the node's chain. The block is first verified by the node
# and then added to the chain.
@app.route('/add_block', methods=['POST'])
def verify_and_add_block():
    block_data = request.get_json()
    block = Block(block_data["index"],
                  block_data["transactions"],
                  block_data["timestamp"],
                  block_data["previous_hash"],
                  block_data["nonce"])

    proof = block_data["hash"]
    try:
        blockchain.add_block(block, proof)
    except ValueError as e:
        return "The block was discarded by the node: " + e.str(), 400

    return "Block added to the chain", 201

# endpoint to query unconfirmed transactions
@app.route('/pending_tx')
def get_pending_tx():
    return json.dumps(blockchain.unconfirmed_transactions)

def consensus():
    """
    Our naive consensus algorithm. If a longer valid chain is
    found, our chain is replaced with it.
    """
    global blockchain

    longest_chain = None
    current_len = len(blockchain.chain)

    for node in peers:
        response = requests.get('{}chain'.format(node))
        length = response.json()['length']
        chain = response.json()['chain']
        if length > current_len and blockchain.check_chain_validity(chain):
            current_len = length
            longest_chain = chain

```

```
if longest_chain:  
    blockchain = longest_chain  
    return True
```

```
return False
```

```
def announce_new_block(block):  
    """
```

```
    A function to announce to the network once a block has been mined.  
    Other blocks can simply verify the proof of work and add it to their  
    respective chains.  
    """
```

```
    for peer in peers:  
        url = "{}add_block".format(peer)  
        headers = {'Content-Type': 'application/json'}  
        requests.post(url,  
                       data=json.dumps(block.__dict__, sort_keys=True),  
                       headers=headers)
```

```
# Uncomment this line if you want to specify the port number in the code  
#app.run(debug=True, port=8000)
```

## 6. TEST CASES:

S.NO	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RATE
1	Create Vehicle Network	Simulation screen created	Simulation screen created	SUCCESS
2	Trusted Authority Key Generation	Generated keys for encryption	Generated keys for encryption	SUCCESS
3	Run Credit Coin Simulation	Allow selected Initiator to send request to nearer vehicle and get reply	Allow selected Initiator to send request to nearer vehicle and get reply	SUCCESS
4	Trace Manager Verification	View all vehicles details	View all vehicles details	SUCCESS
5	Computation Time Graph	Get graph for each vehicle request, reply sending and verification time	Get graph for each vehicle request, reply sending and verification time	SUCCESS

## 7.SCREENSHOTS

To run project double click on 'run.bat' file to get below screen

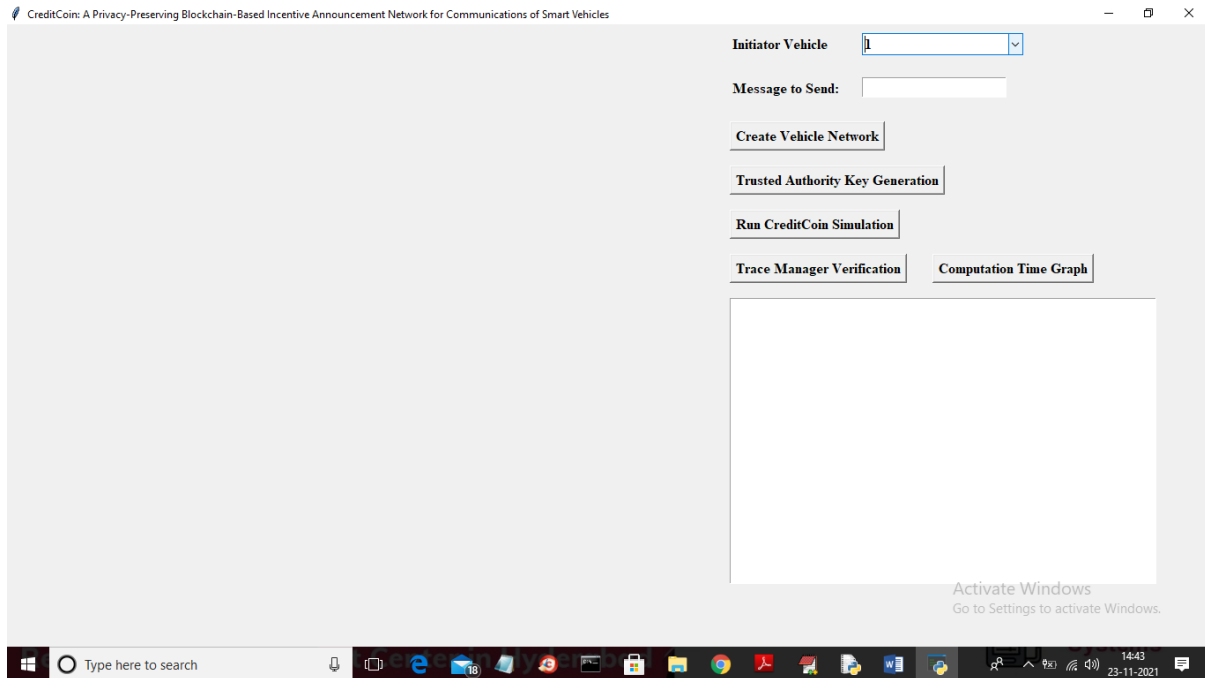


Fig.7.1 Initiator Vehicle

In above screen first drop down box contains all vehicles ID's and then you can enter message to send to other vehicle



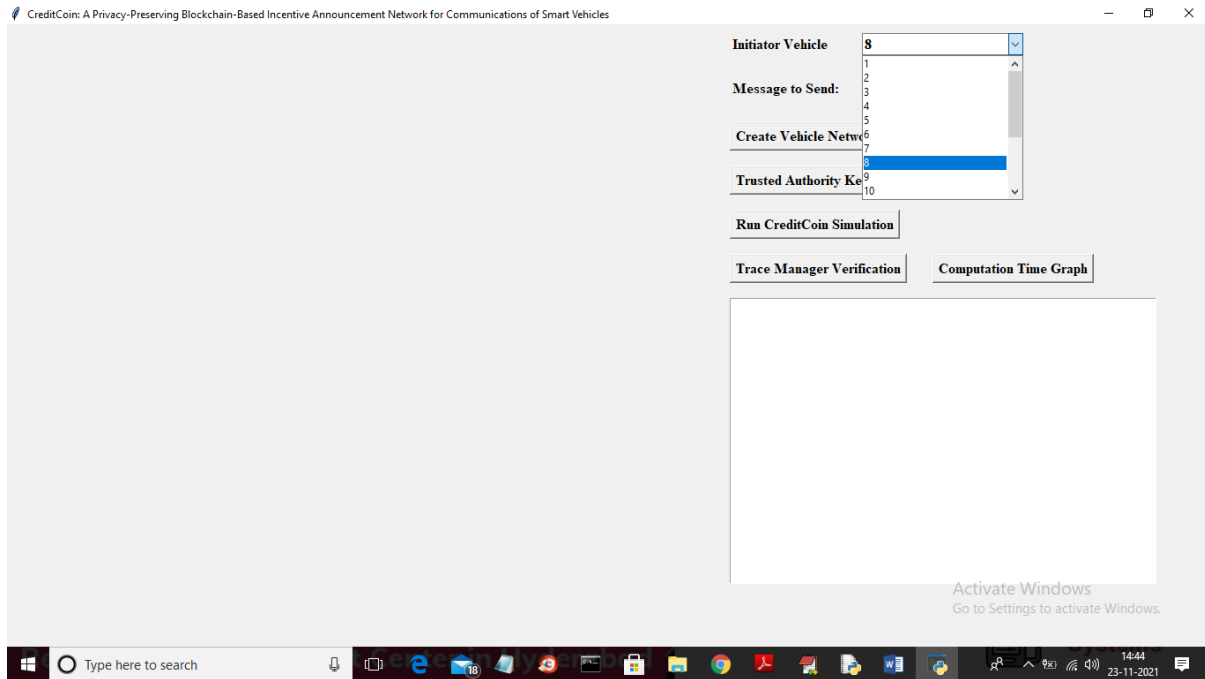


Fig.7.1.1 Initiator vehicle Number

In above screen from drop down list I selected INITIATOR vehicle id as 8 and then enter some message to send to other vehicle

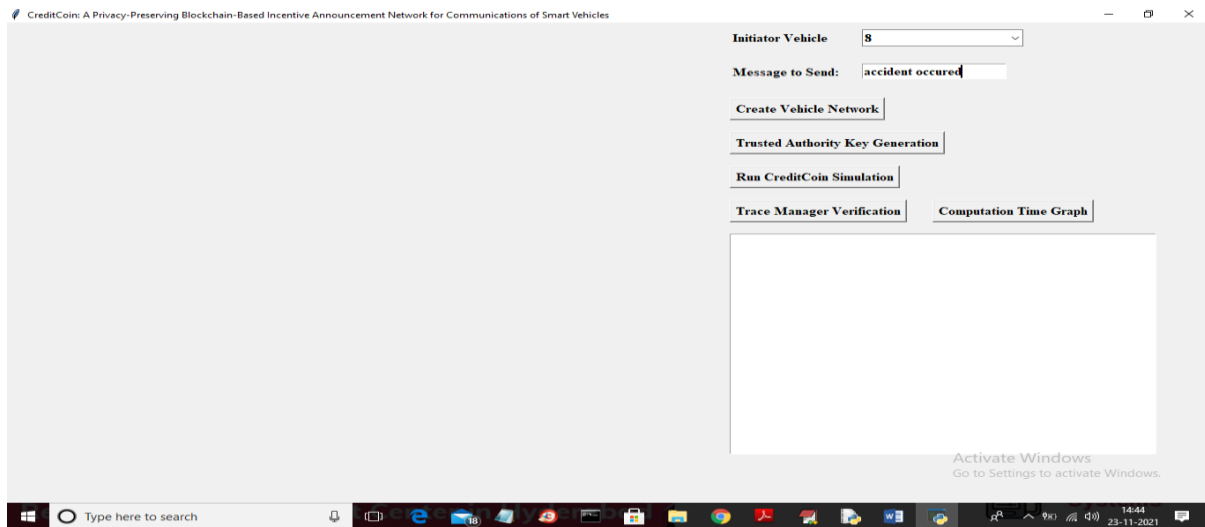


Fig.7.2 Sending Message

In above screen I entered some message and now click on 'Create Vehicle Network' button to create simulation screen

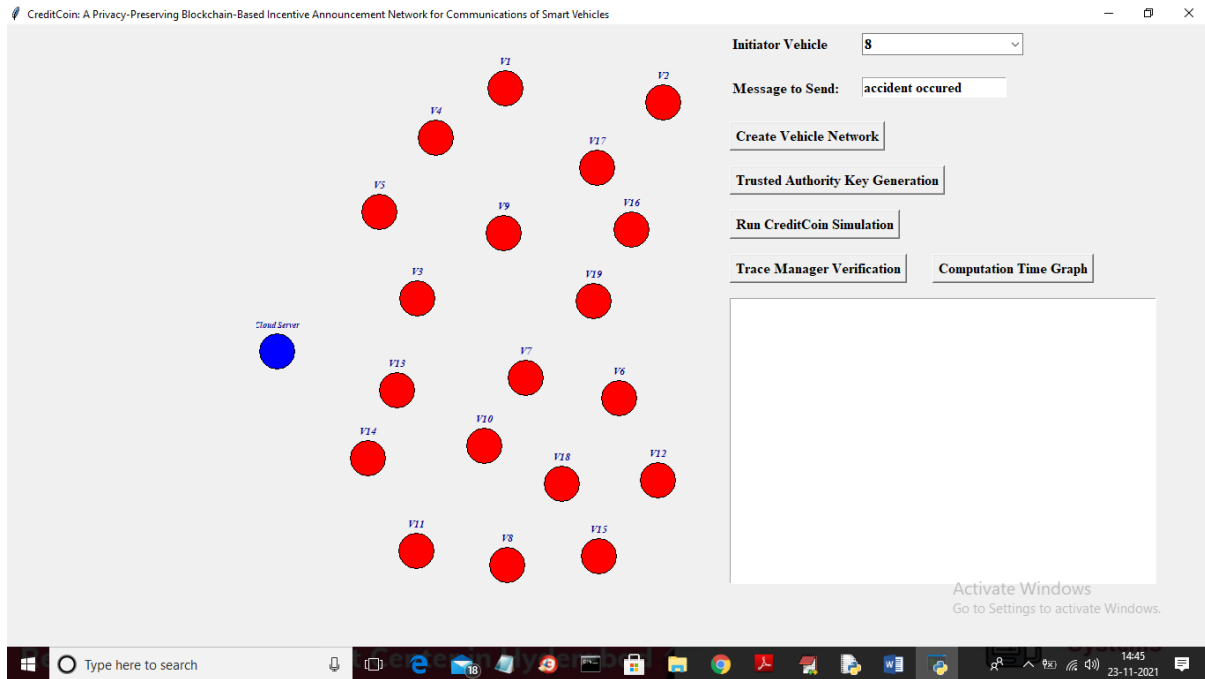


Fig.7.3 Creation Of Vehicular Network

In above screen all red colour circles are vehicles and blue color circle is the Cloud Server and now click on 'Trusted Authority Key Generation' button to generate keys for encryption

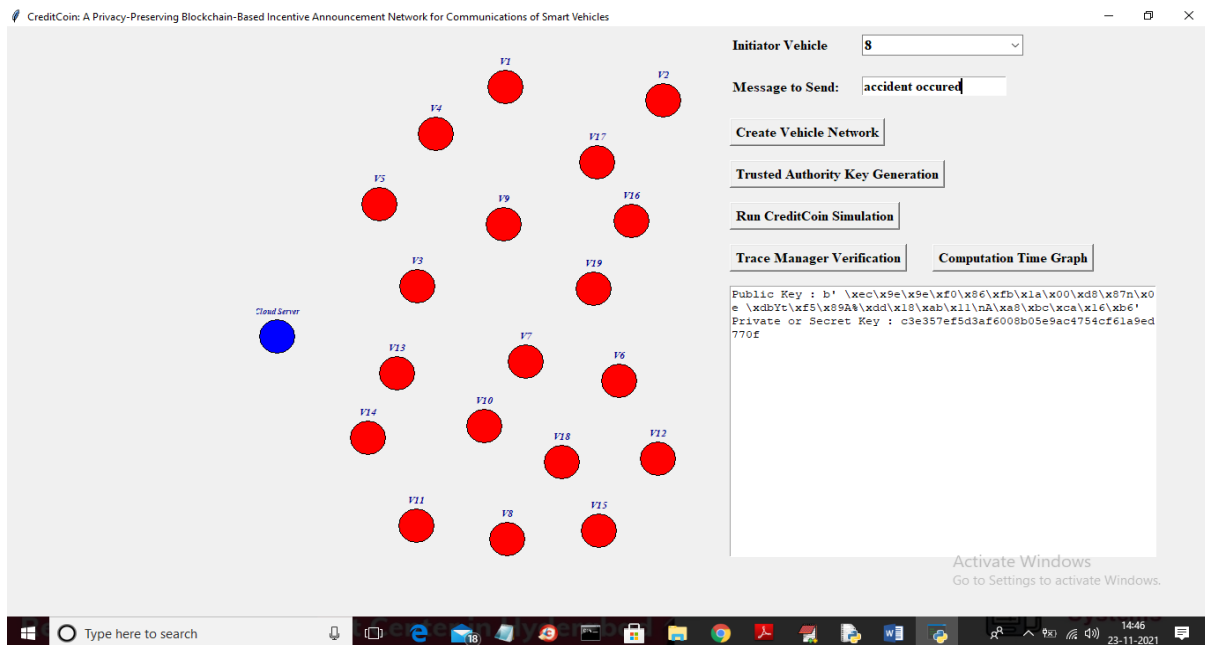


Fig.7.4 Key Generation

In above screen in text area we can see keys are generated and now click on ‘Run Credit Coin Simulation’ button to allow selected Initiator to send request to nearer vehicle and get reply. All this data will be stored in Blockchain so it may take few seconds time to show below output

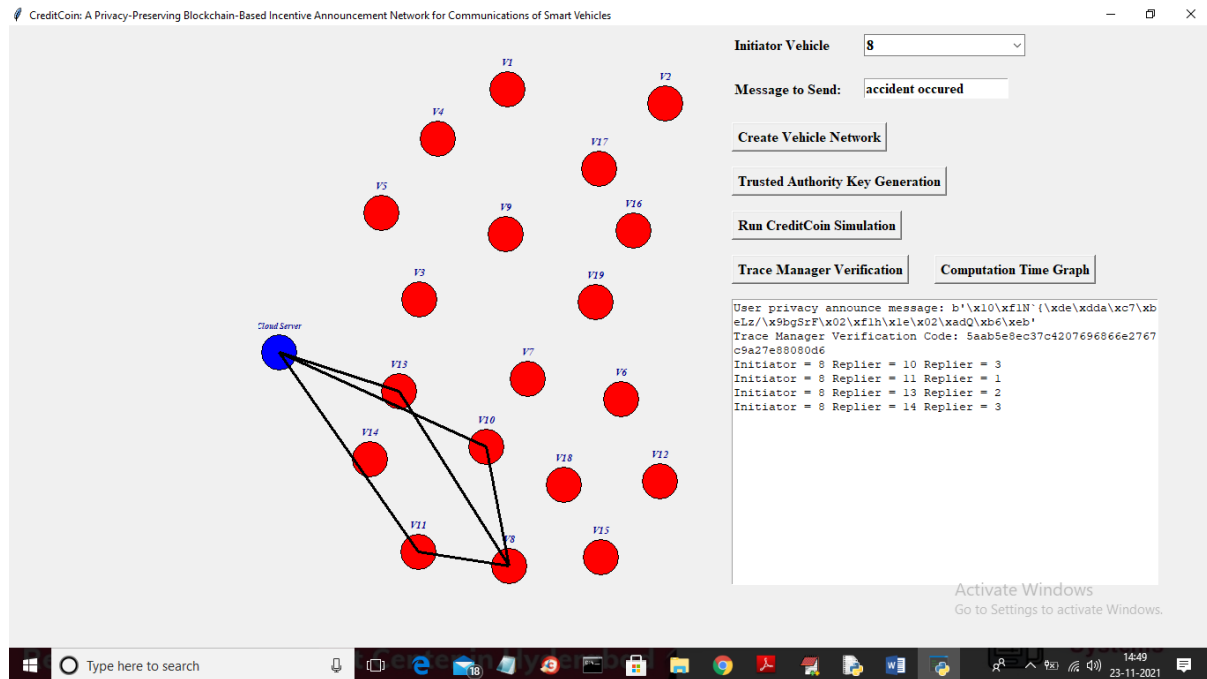


Fig.7.5 Credit Coin Simulation

In above screen we can see INITIATOR and repliers are exchanging announcement between each other and cloud server and in text area we can see which initiator sending request to which replier and all this data will saved in Blockchain and to read or verify data click on ‘Trace Manager Verification’ button to get below output

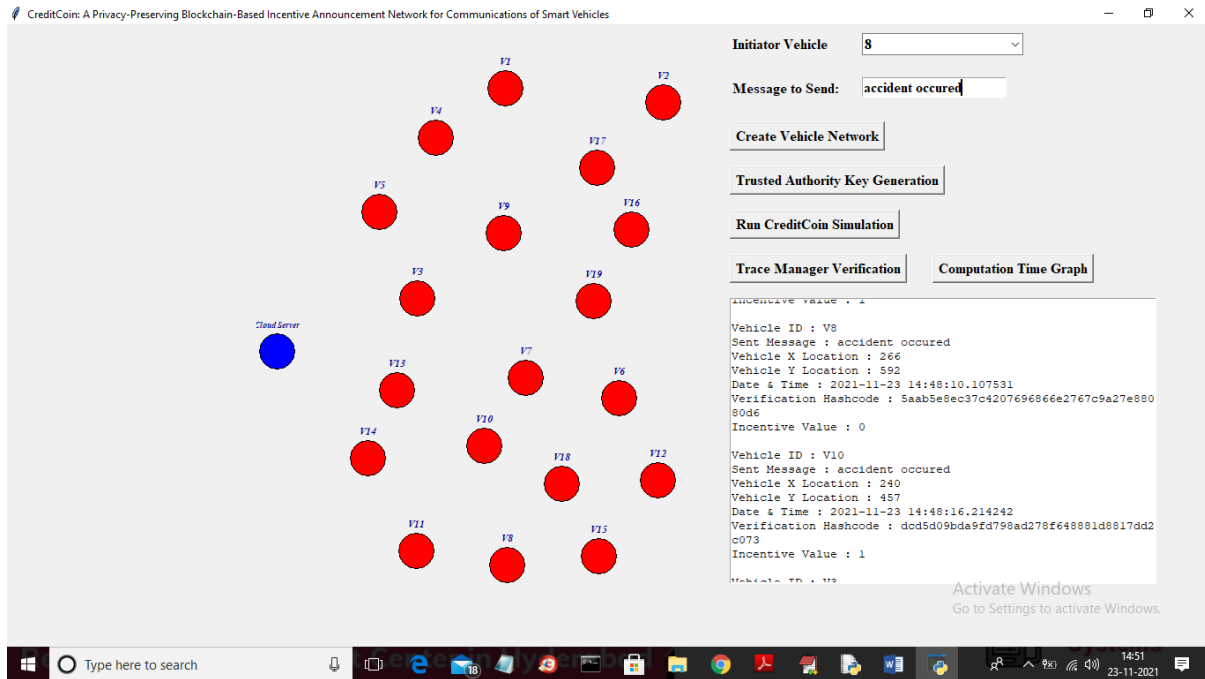


Fig.7.6 Trace manager

In above screen in text area we can see decrypted vehicle ID involved in announcement with their X and Y location, Date time and verification hashcode and incentive values and if hashcode mismatch then verification will be failed. You can scroll down above text area to view all vehicles details like below screen

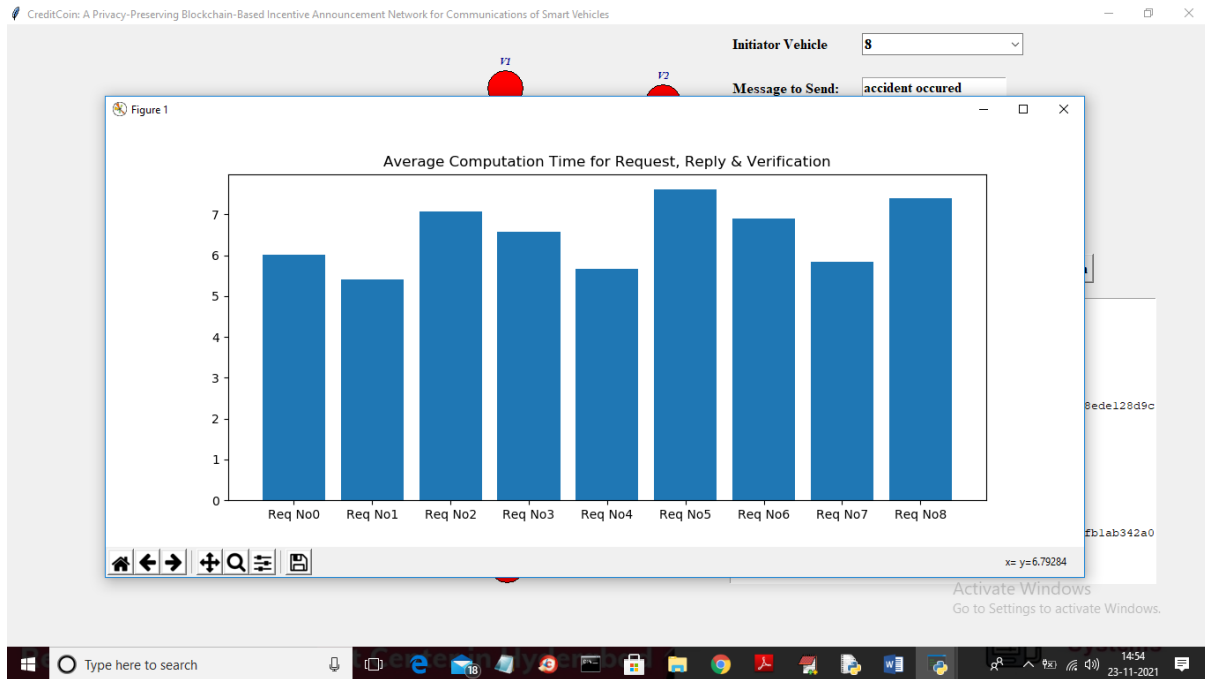


Fig.7.7 Computation Graph

In above screen click on 'Computation Time Graph' button to get below graph for each vehicle request, reply sending and verification time. In above screen x-axis represents request no and y-axis represents computation time require for sending request, reply and verification.

## 8. CONCLUSION

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 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. 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.

## **9. FUTURE ENHANCEMENTS**

In the future, the enhancement of the proposed Vehicular Announcement Network (VANET) project can focus on addressing the challenges related to anonymous message forwarding and user motivation. To ensure the reliability of messages while maintaining anonymity, advanced cryptographic techniques and privacy-preserving protocols can be integrated into the VANET system. This may involve the use of homomorphic encryption, zero-knowledge proofs, or other privacy-enhancing technologies to authenticate and verify messages without revealing sensitive user information. By implementing robust privacy measures, the project aims to strike a balance between maintaining the confidentiality of user data and ensuring the trustworthiness of vehicular announcements in the network.

Furthermore, to incentivize active participation and message forwarding among users, a gamification approach can be introduced. By incorporating elements of competition, rewards, and recognition into the VANET system, users can be encouraged to actively engage in forwarding announcements, contributing to the overall efficiency and reliability of the network.

## 10. 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. Szydło, “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.