CHAPTER 1

INTRODUCTION

1.1 General Introduction

Smart traffic systems have become an essential part of modern-day transportation, providing numerous benefits such as reducing traffic congestion, improving safety, and optimizing traffic flow.

However, with the increased connectivity and reliance on technology, smart traffic systems are also vulnerable to cyber attacks that can compromise the safety and privacy of the system's users. To address these security concerns, researchers and developers have been exploring the use of blockchain technology to develop secure communication methods in smart traffic systems.

Blockchain's decentralized and immutable nature provides a secure and transparent platform for data sharing and communication among the various components of a smart traffic system. This project will explore the development of secure communication methods using blockchain technology in smart traffic systems, including the benefits and challenges of implementing such systems, and the potential for wider adoption in the future.

1.2 Problem Statement

The problem this project aims to address is the lack of trust and efficiency in traditional methods of collecting and managing data on traffic conditions. There is often a lack of transparency in how this data is collected and managed, leading to a lack of trust among stakeholders. Even though VANET is a turning point in vehicular communication, it accompanies many security risks and problems. In travel comfort applications, security issues are not often considered because cooperative driving is typically assumed. Moreover, existing electronic voting systems are often centralized, making them vulnerable to hacking and other forms of cyber attacks, which can compromise the integrity of the data collected. This can be particularly problematic when it comes to collecting data on traffic conditions, where accuracy and timeliness are crucial for effective traffic management and ensuring road safety.

1.3 Significance of the Problem

Due to the open wireless access medium, the security and privacy of this information become quite critical in VANETs. The attackers could capture, intercept, alter, replay and delete the traffic-related information and could compromise the security of VANETs. If an attacker is able to capture important data it would lead to many unforeseen situations such as hours-long traffic jams which could lead to disturbance for ambulances, police vehicles and other important vehicles. Apart from this it could lead to incorrect vehicle data which could lead to chaos on roads. Then there is private information leakage , personal data and important data which no other vehicle should know.

The significance of the problem addressed by this project lies in the critical importance of accurate and timely data on traffic conditions for effective traffic management and road safety. Traffic congestion, accidents, and other road-related issues can have severe consequences, including loss of life, injury, and economic losses.

Currently, many cities and transportation agencies rely on manual methods of data collection, such as surveys and physical inspections, which can be time-consuming, expensive, and prone to human error. This can lead to inaccuracies in the data collected, delays in decision-making, and ultimately, negative impacts on road safety and traffic flow.

The lack of trust, transparency, and efficiency in traditional methods of collecting and managing data on traffic conditions further exacerbates the problem. This can create a lack of confidence among stakeholders, including government agencies, transportation operators, and the general public, in the data collected and the decision-making process.

By developing a decentralized e-voting system using blockchain technology, this project can address the limitations of traditional data collection and management methods. The use of blockchain technology can provide a secure, transparent, and efficient way to collect and manage critical information on traffic conditions, ensuring the integrity of the data collected and promoting trust and accountability among stakeholders.

Ultimately, the significance of this problem lies in its potential impact on improving road safety, reducing traffic congestion, and promoting more efficient transportation systems. By addressing the limitations of traditional methods of data collection and management, this project can contribute to a safer and more sustainable future for communities worldwide.

1.4 Brief Description of the Solution Approach

The main objective of the project is to develop an efficient framework for establishment of a secure communication channel among vehicles in a shared network. This can be achieved by verifying a vehicle when, or even before, it tries to connect to another vehicle. After establishment of a secure connection, blockchain technology can be used to provide a secure, transparent, and efficient way to collect and manage critical information that can ultimately lead to improved road safety and traffic flow.

The solution approach to the development of secure communication methods in smart traffic systems using blockchain technology involves the integration of various components such as cryptography, distributed ledger technology, and consensus mechanisms. The goal is to ensure secure and transparent communication between the various entities within the smart traffic system, such as vehicles, traffic signals, and other infrastructure.

One key aspect of this solution approach is the use of cryptography to secure communication channels and data. This involves the use of encryption and digital signatures to ensure the authenticity and integrity of the data being transmitted. Additionally, the use of smart contracts and distributed ledger technology allows for the automation and transparency of transactions, reducing the risk of fraud and errors.

CHAPTER 2

BACKGROUND STUDY

Various studies have investigated the importance and advantages of implementing secure communication techniques among vehicles to address real-world challenges. Incorporating a secure communication model between vehicles can improve the accuracy of data stored in the system and enhance the performance of existing models. We have examined a number of research papers and utilized their models to develop our own communication model. Some of the research papers we have reviewed include those that focus on secure communication protocols, data sharing schemes, and group key management schemes for vehicular ad hoc networks (VANETs).

2.1 Literature Survey

A survey paper[1] examined approximately 75 security schemes that use blockchain technology for vehicular networks. The survey analyzed these schemes from different viewpoints, including their application, security, and utilization of blockchain technology.

1. The perspective of the application was centered around different uses of secure blockchain-based vehicular networks such as transportation, parking, resource sharing, and exchanging data.
2. The security perspective focused on security requirements and attacks.
3. The focus of the blockchain perspective was on the types of blockchain platforms, consensus mechanisms, and blockchain implementations.

In addition to reviewing various research papers on secure communication methods for vehicular networks, the authors also compiled a list of popular simulation tools and blockchain applications relevant to this area.

One of the papers[2] proposed a proof-of-event consensus concept that utilizes roadside units to collect traffic data and passing vehicles to verify the accuracy of event notifications. This method addresses key VANET requirements such as authentication, integrity, non-repudiation, privacy, and efficiency.

Another paper suggested equipping vehicles with wireless communication devices called onboard units (OBUs) that contain a processor, storage, network device, and sensors.

On-board units (OBUs) use a specialized communication protocol known as Dedicated Short-Range Communication (DSRC) to communicate with other OBUs or Roadside Units (RSUs).

Anonymity is a crucial aspect of protecting the privacy of vehicles in VANETs, and anonymous authentication is a commonly used method to achieve this. However, the issue of scalability in data storage for VANETs has been a challenge. In order to tackle this issue, scholars put forth DSSCB, a system for sharing and storing data securely, that employs a consortium blockchain. The use of a consortium blockchain helps to improve scalability and overall efficiency. The integrity of data is maintained through the use of digital signatures when data is uploaded by a vehicle.

A proposed solution to enhance vehicle security involves three main components: trusted cloud service providers, Roadside Units (RSUs), and vehicles. The vehicle's registration process involves submitting authentic identity data to the RSU. Once validated, the RSU encrypts the data and sends it to the cloud service provider. The cloud service provider assesses the credibility of the vehicle information and shares it with other RSUs via a consensus algorithm. The contract nodes evaluate the vehicle's credit record and rating to determine trustworthiness of the new application node.

TangleCV is a new decentralized approach for secure message sharing among connected vehicles, which utilizes a Proof of Work (PoW) mechanism to validate new transactions against two previous ones. The aim is to improve efficiency and scalability while maintaining data accuracy. The system also accounts for priority vehicles, such as ambulances, police or military vehicles, and vehicles carrying crucial goods. The research findings indicate that TangleCV has demonstrated improved performance in terms of information correctness, which can benefit various applications in the connected vehicles' environment.

The B2VDM is a blockchain-based architecture designed for secure management of vehicular data at RSUs. It utilizes a consensus algorithm called "proof of existence" to ensure reliability. When an access request is made at an RSU, the system checks the blockchain for the existence of a corresponding transaction. Any redundant transactions are eliminated, and only authorized applications can access the data. This eliminates conflicts that may arise among multiple service providers and ensures privacy for vehicular data.

It is important that the actual identity of a vehicle remains hidden from both roadside units (RSUs) and other vehicles. Additionally, it is essential to prevent any unauthorized individual from accessing the real identity of the vehicle.

In a scenario where an event takes place, several vehicles transmit messages regarding the event to the closest Roadside Unit (RSU). The RSU then evaluates the reputation of the vehicles that initiated the messages to determine the authenticity of the reports. The resulting reputation values are then recorded in a block. This approach employs a trust calculation technique that utilizes logistic regression to increase the precision of the reputation score in detecting malicious vehicles.

The paper mentioned in [5] utilized smart vehicles equipped with sensors to collect and share information about traffic events, which can be useful for improving the transportation system. However, due to the increasing threat of cyber-attacks, securely storing and sharing the event messages collected through VANETs has become a significant challenge. To address this, the authors proposed using blockchain technology to create a decentralized, transparent, and robust database, which can form the basis of a secure traffic event management protocol. They adopted the Proof of Work (PoW) consensus mechanism to ensure the integrity and reliability of the protocol.

Proof of Work (PoW) is a well-known mechanism in blockchain technology that provides high fault tolerance and resilience against attacks from adversaries. It ensures that the blockchain network remains secure unless more than half of the total computational power of the mining nodes is controlled by malicious entities. In the context of vehicular networks, Road Side Units (RSUs) provide various services and applications to the vehicles. They are located at the sides of the roads and serve as edge nodes to store the collected data.

In a research paper[3], the authors have proposed using blockchain technology to collect and display road traffic information securely in a network of connected vehicles. They have developed a prototype to demonstrate the effectiveness of their approach and its resilience against privacy attacks. The paper aims to build upon existing research on traffic data gathering and consumption and offers a private solution that enables road management while protecting user privacy.

A research paper[4] has proposed using Blockchain to store trust-related information for interactions between different entities. However, existing trust management systems that use Blockchain lack a way to verify the interactions on which the trust score is based. The authors of the paper suggest a new framework that allows independent trust providers to implement different trust metrics using a shared set of trust evidence, resulting in individualized trust values.

Previous proposals for secure communication in vehicular networks have used geo-location data as a means of verifying interactions, but these methods are limited to centralized systems and do not support trust calculation by multiple providers. Our proposed architecture uses blockchain technology to ensure provable interactions between entities, without relying on a centralized third-party entity for trust management. This approach allows for a high level of confidence in trust management, as the authenticity of interactions can easily be verified.

2.2 Comparative analysis

S. no	**Name of paper**	**Year, problem statement**	**Inputs considered in smart contract**	**Consensus protocol**	**Methods used**	**Simulations carried**	**How the performance was calculated**
1.	Blockchain-Based Traffic Event Validation and Trust Verification for VANETs	2019, accident detection	speed, location, heading, signature of the synopsis result and public-key certificate of the vehicle	Trust-based proof of event	Public key infrastructure (PKI) -based model two-phase transaction Verifier is chosen randoml	NS-3 simulator	1.Compared PoE, PoA and PoW by varying the number of RSUs and measuring sync time. 2. Success rate vs no. threshold for min no. of validations

					y		
2.	Using Blockchain to Enhance and Optimize IoTbased Intelligent Traffic System	2019, lane property right acquisition	hash, identity of vehicle, speed of vehicle, token payment (emergency level), previous block hash	-	-	-	-
3.	VeriBlock: A Blockchain-Based Verifiable Trust Management Architecture with Provable Interactions	2022, notify about heavy traffic or a roadside accident	time, location and classification of accident	Trust-base, proof of interaction	1. Trust all nodes - then calc %age and compare with threshold 2. Same as above but filter on basis of time and location 3. Weighted (70/30)	Python 3.9 and the Web3 library to interact with the private Ethereum network	average time taken for a user to submit a transaction and have it recorded on the blockchain average time for a car to receive information from a request

					result of above 2		
4.	A Blockchain based Incentive Provisioning Scheme for Traffic Event Validation and Information Storage in VANETs	2020, secure, avoid initiators acting selfishly, incentive mechanism to encourage participation, store large amount of data	Trusted authority signature and status Event address and info Signature value and initiator Balance Incentive value Reputation value	weighted ranking algorithm Reputation based trust (correctness of an event it signs or initiates) , proof of authority algorithm to detect the fake news. The RSUs perform the consensus using Proof of Work (PoW) mechanism.	IPFS: a distributed storage mechanism weighted ranking Algorithm Provide monetary incentive (transaction stored in blockchain)	Python, Oyente tool (security of smart contract), smart contract is tested on Remix IDE, Ganache and Metamask	transaction cost and computation time (transaction cost is the total amount of gas that is required to send the data to the blockchain network) comparison of the proposed scheme with the logistic regression scheme

5.	A Blockchain-Based Approach to Track Traffic Messages in Vehicular Networks	2022, Sharing traffic information securely	-	proof of work (PoW) and the practical Byzantine fault tolerance (PBFT) consensus algorithms.	Traffic events transactions are generated according to a Poisson distribution. events validation model, credibility scoring module based on historical data from blockchain	implemented in C++ and connected to NS3.	Throughput (no. of traffic events that the blockchain can verify per second) Latency (average amount of time required for an event to be written in blockchain)
6.	A Secured Message Transmission Protocol for Vehicular Ad Hoc	2020, blockchain-based Secured Cluster-based MAC (SCB-MAC) protocol	safety message which contains emergency information and requires Strict	-	-	Ganache, metamask, Lightweight NPM Server	Throughput and packet dropping rate

	Networks		Delay Requirement (SDR)				
7.	Blockchain-based anonymous authentication for traffic reporting in VANETs	2022, certificateless message signature Algorithm, adaptive threshold multi-signature mechanism, T-Coin	target hash value, random number and timestamp	-	-	Java	Vehicle number vs avg computation time
8.	Blockchain-Based Secured IPFS-Enable Event Storage Technique With Authentication Protocol in VANET	2021, protocol for secure communication	User identity, RSU identity, Eth address of use, Eth address of RSU, Event	Proof of Work (PoW)	-	JavaScript VM, IPFS	transaction cost (in terms of GAS)
9.	Competitive-blockchain-base	2022, automatic parking	Price of parking, capacity of	-	consensus mechanis	Rust (due to its particular memory	-

	d parking system with fairness constraints	system in which vehicles are allocated among several competitive parking areas	parking lot – rate per hour; – closing hours; – key deposit;		m to manage the system modifications.	management system, allows building fast, efficient and secure applications that can be executed on integrated devices using a small amount of resources.)	
10.	When Proof-of-Work (PoW) based blockchain meets VANET environments Conference	2021,blockchain technology for a decentralized, transparent and robust database , secure traffic event man-agement protocol.	Event report,threshold, memepool(valid events), Input time	Proof of Work (PoW)	events' trustworthiness, blockchain's reliability and security. ,poisson distribution, Schnorr signa-ture	NS-3 simulator	No of events,malicious vehicles, Threshold,event latency

11.	A Novel Multifaceted Trust Management Framework for Vehicular Networks	2022 , TMS Trustworthiness,stable interconnectivity,block based correctness.	Trust value table, RSU event,dely,trust, distance x`	Proof of event	Clustering Algorithm, Blockchain.precision,Recall, Security module in Blockchain	OMNET++, SUMO ,	Trust Models, Accuracy with confusion matrix
12.	Blockchain and Cooperative Intelligent Transport Systems: Challenges and Opportunities	2022 impact of the definition of Blockchain-based solutions in C-ITS. Cooperative Intelligent Transport Systems	-	Proof of location, Proof of Event	Reputation system, Incentive mechanism	-	-

Table 1: Comparison of different research papers

CHAPTER 3

REQUIREMENT ANALYSIS AND SOLUTION APPROACH

3.1 Overall Description of the Project

In this project we designed and implemented a secure vehicle verification protocol for VANETs. The components of the authentication framework are RTA (Regional Trusted Authority) , RSUs (road side units) and different vehicles. This framework was implemented via simulation using Python. Time analysis was done for key generation and transmission in the simulation, with testing and exception handling.

To promote transparent and tamper-proof communication, blockchain technology was implemented. For instance, consider the poll conducted by Google Maps to identify roadblocks, accidents, or traffic congestions. However, such a centralized approach is vulnerable to cyber-attacks, which may lead to data tampering. Additionally, this approach requires a significant amount of infrastructure and lacks transparency. The accountability, trust, and integrity of the system entirely rely on the company managing the centralized data. These issues can be addressed by adopting blockchain technology, which offers solutions to all these problems.

There are three main components of the project:

1. Secure vehicle verification protocol: The framework is illustrated as follows:

Step no.	Operation	Data transferred	Description
Step 0	RTA → Vehicle	Key pair for vehicle (PKv, SKv)	RSU provides a vehicle with its asymmetric key pair at border (toll booth at state borders)
Step 1	RTA → RSU	Key pair for RSU (PKr, SKr)	This key pair is refreshed every 10 min or so to prevent brute force attacks
Step 2	RSU → *	Rid, PKr, SIGr	RSU is broadcasting its details for nearby vehicles to connect through

Step 3	Verify SIGr	-	Decrypt SIGr with PKr, and compare result with Rid
Step 4	Vehicle → RSU	Rid, PSv, SIGv	Vehicle develops SIGv and PSv for its verification by RSU and RTA
Step 5	RSU decrypts PSv with SKr	-	Decrypting PSv gives the vehicle ID Vid
Step 6	RSU → RTA	Vid	Vid is sent to the RTA through a secure communication channel to preserve integrity
Step 7	RTA → RSU	PKv	The RTA checks its table for corresponding PKv of vehicle with ID = Vid
Step 8	Verify vehicle	-	RSU decrypts SIGv with PKv and compares PSv. If the PSv matches, the vehicle is legitimate, thus authenticated
Step 9	RSU → *	PSv	Every vehicle in VANET is transmitted the PSv of verified vehicle

Table 2: Procedure of secure vehicle verification Protocol

PK : Public key

SK : Private key

Rid : RSU ID

SIGr : Signature by RSU = encrypt(Rid, SKr)

PSv : Pseudonym of vehicle = encrypt(Vid, PKr)

SIGv : Signature by vehicle = encrypt(PSv, SKv)

2. Simulation: The simulation was aided by Jianshan Zhou's VANET Simulation project on Github [12], which was modified to fit our needs. The original simulation project was developed in Python to explore implementation of a simple epidemic-based routing mechanism designed to support reliable message dissemination in vehicular ad-hoc networks. The project also measures propagation distance of message and ratio of message carriers wrt time.

3. Decentralized Road Event Voting: A voting feature was implemented through which a user can report a road event to other users, who can then login and verify the event. Verification of an event uses Proof of Reputation consensus mechanism. Based on the consensus, total votes are calculated and the result of the poll is determined, which can be viewed on another webpage.

3.2 Requirement Analysis

3.2.1 Software Requirements

- Any OS that will run Python compiler and has GUI (Desktop edition)
- Language: Python 3.11
- HTML 5
- DB SQLite 3
- Python libraries:

Vehicle verification framework:

- socket module
- numpy==1.19.5
- tracemalloc module
- rsa == 4.9
- cvs==1.0

Blockchain voting system:

- Django Web Framework 3.0.3
- hashlib==20081119

3.2.2 Hardware Requirements

- Computer Processor: Intel i3 or higher
- RAM: 2GB or more
- Network Interface Card for Socket programming

3.3 Solution Approach

3.3.1 Secure vehicle verification protocol

A Regional Trusted Authority (RTA) is present for each region, say for each state. Within a region, multiple Road Side Units (RSUs) are present. These RSUs monitor passing vehicles and broadcast its public key. Any vehicle wanting to initiate a connection with another vehicle (in the form of VANETs) must verify itself first to the nearest RSU.

For verification, the framework heavily relies on public-key cryptography, or asymmetric cryptography. The RSA algorithm helps to generate public and private key pairs. These keys can be used for:

1. **Encryption** (by using public key of receiver)/**decryption** (by using private key of receiver) to preserve the confidentiality of a message
2. **Digital signature**: by encrypting the message using the sender's private key to verify the authenticity of the message

The RTA, RSU and vehicle interact with each other, as illustrated in table 2, to verify the vehicle. It is assumed that RTA and RSU communicate through a secure communication channel. Yet no confidential data, such as private keys of vehicles are sent over the channel. Only transmitting Vid and key pair of RSU required the need of an established secure communication channel between RSU and RTA.

3.3.2 Simulation

The Python simulation is based on a simple epidemic-based routing mechanism designed to support reliable message dissemination in vehicular ad-hoc networks. The simulation can be carried out under different mobility by adopting different distributions of vehicle speed and inter-vehicle space, which provides several typical traffic scenarios for studying the epidemic routing mechanism. The basic mobility model used here is based on the well-known intelligent driver model (IDM) and the default mobility parameters involved are set according to existing literature. The simulator was aided by Jianshan Zhou’s Github project, a Research fellow in Beijing Key Laboratory. The original project was cloned and modified to fit the needs of this project.

In traffic flow modeling, the intelligent driver model (IDM) is a time-continuous car-following model for the simulation of freeway and urban traffic. It was developed by Treiber, Hennecke and

Helbing in 2000 to improve upon results provided with other "intelligent" driver models such as Gipps' model, which loses realistic properties in the deterministic limit.

The followings things were measured over the simulation:

1. *Time for key generation and transmission in the simulation*: This will help in determining the feasibility of the framework in real life scenarios
2. *Plots for different mobility scenario*: To observe the implementation in different traffic conditions

The mobility scenarios are:

```
scenario_flag = "Freeway_Free":
    self.headway_random = lognorm(0.75, 0.0, np.exp(3.4))
    meanSpeed = 29.15 #m/s
    stdSpeed = 1.5 #m/s
```

```
scenario_flag = "Freeway_Rush":
    self.headway_random = lognorm(0.5, 0.0, np.exp(2.5))
    meanSpeed = 10.73 #m/s
    stdSpeed = 2.0 #m/s
```

- headway_random: Headway is the distance or duration between vehicles in a transit system measured in space or time. Headway_random is used to induce randomness as there is no interaction between the arrival of two vehicles
- meanSpeed: average speed of all the vehicles in the simulation
- stdSpeed: Standard deviation of the vehicle speed

3.3.3 Blockchain Implementation

A functionality for voting has been introduced that enables a user to submit a traffic event for other users to review and confirm by logging into the system. Each set of votes are stored as blocks in a blockchain.

The Proof of Reputation consensus mechanism works by assigning each participant in the network a reputation score based on their behavior within the network. This score is calculated based on various factors, such as their past behavior, their contributions to the network, and their interactions with other participants. Our system assigns reputation scores to each participant in the network based on their honesty within the system. These reputation scores are used to determine each participant's voting power in the decision-making process, with higher reputation scores resulting in greater influence.

Besides reputation, randomness also plays a role in filtering voters. A random set of voters are selected whose reputation adds up to the desired capacity of the system (vehicleCap). This is done in consideration of cases when a highly trusted voter may input wrong votes. In summary, the consensus ensures that only a limited number of voters are selected randomly based on their reputation to participate in the voting system. The reputations of voters are adjusted based on their voting behavior, and the system strives to maintain a balance between randomness and reputation to ensure the integrity of the voting process.

To run the project locally:

1. Clone the project locally:
   `git clone`
`https://github.com/Corbe30/Development-of-Secure-Communication-Methods-in-Smart-Traffic-Systems/upload`
2. Install dependencies (if not installed already):
   `pip install django`
3. Update path variable in watcher.py to that of vehicleList1.csv.
4. Run Django server
   `python3 manage.py runserver`

vehicleList1 file is generated by scenario.py, and contains the list of all the vehicles successfully verified by the RSU and RTA. To share the blockchain and ledger with others on the network, a node can use the `shareBlockchainLedger()` and `receiveBlockchainLedger()`functions.

CHAPTER 4

MODELING AND IMPLEMENTATION DETAILS

4.1 Design Diagrams

4.1.1 Use Case Diagrams

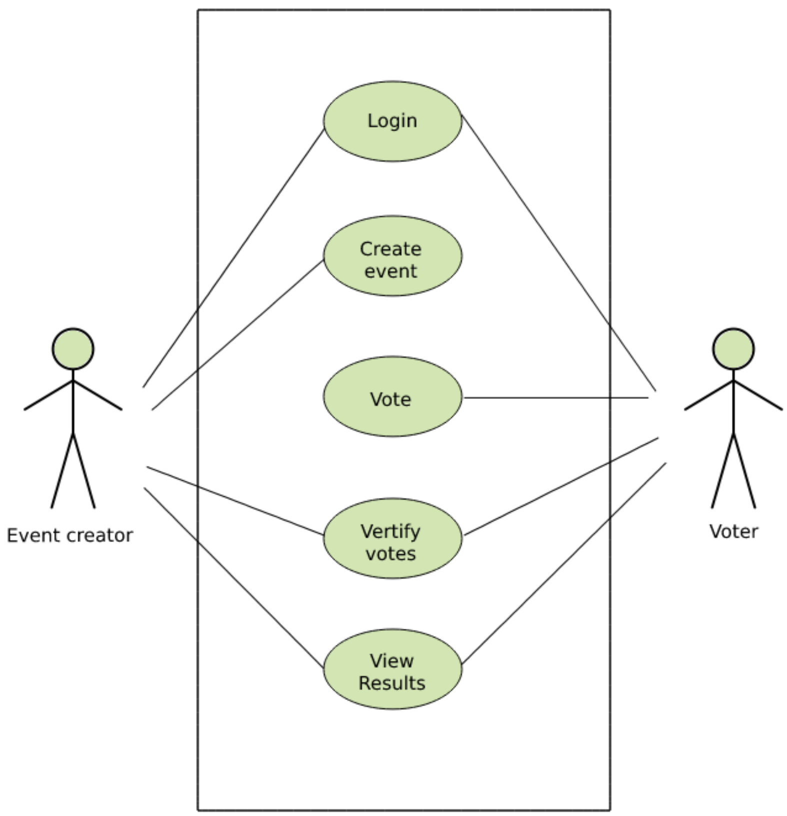


Figure 1: Use case diagram of e-voting in blockchain system

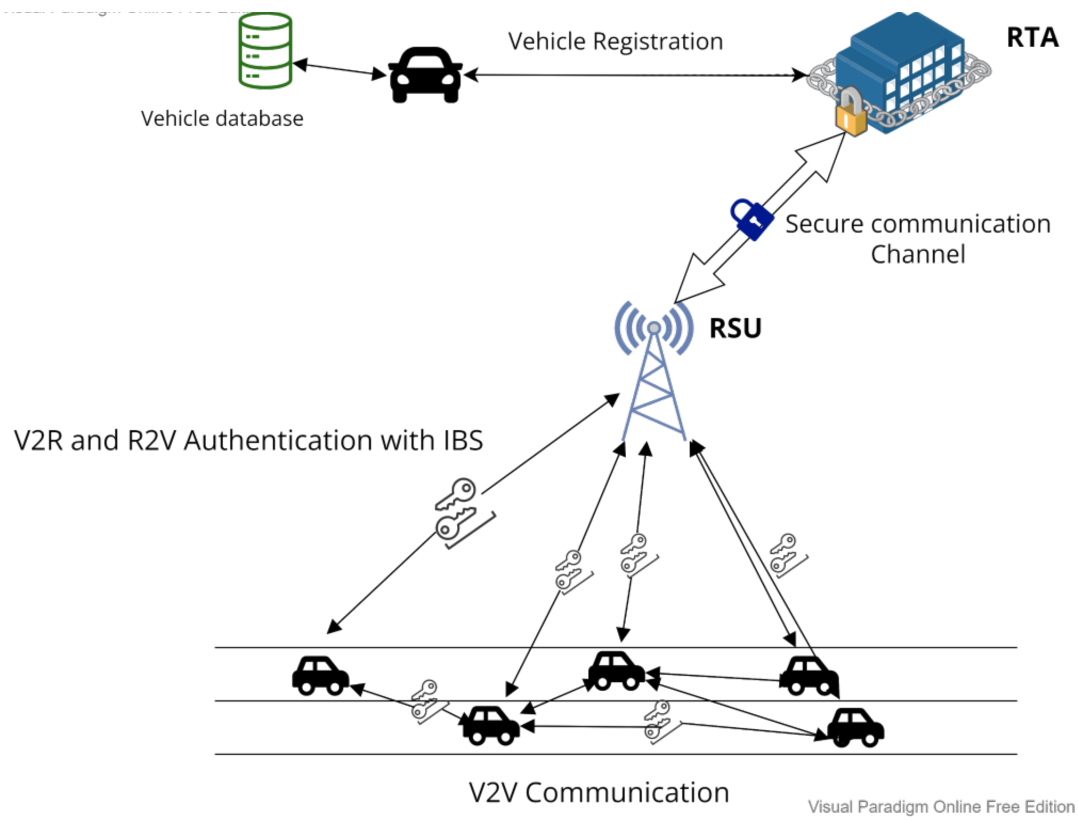


Figure 2: Depicts overall process of secure communication

4.1.2 Control Flow Diagram

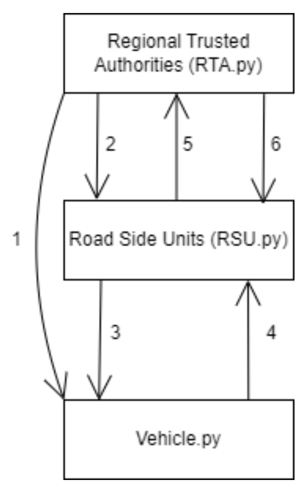


Figure 3: Flow of data transferred among different files

S no. in diagram	Data transferred
1	PKv, SKv
2	PKr, SKr
3	Rid, PKr, SIGr
4	Rid, PSv, SIGv
5	Vid
6	PKv

Table 3: Which data is transferred among different files

4.2 Implementation Details and Issues

1. *RSAfuntions.py*

Create rsa encryption, decryption and verification codes. rsa library in python was used for creating a set of reusable code which will be used throughout the project for security and verification purposes. SHA-1 algorithm was used for hashing.

We have created 4 functions:

i. **RSAencrypt** - used to encrypt the message sent between vehicles, RSU and RTA.We have used the Public key of the receiver to encrypt the message.
ii. **RSAdecrypt** - used to decrypt the message sent between vehicles ,RSU and RTA.The receiver uses its Private/Secret Key to decode the cryptographically secure message sent to it.
iii. **RSAsign** - It is used to sign the identity of the sender. The sender hashes the message with its Private Key. A hashing algorithm is used to hash the message and the digital signature is sent.
iv. **RSAverify** - It is used to verify the identity of the sender. The receiver verifies the signature with the help of the public key of the vehicle and verifies whether the message is received from a valid sender or someone impersonating as the sender.

2. *Vehicle.py*

a. Create vehicle parameters and assign it to the vehicle:

Vehicle.py creates the basic identity of the vehicle by assigning it a vehicle ID , which direction will the vehicle be on the lane , which lane the vehicle will be on, what will be the position of the vehicle , what will be the vehicle speed and its acceleration.

b. Get vehicle public-private key pair from RTA

Vehicle.py requests a unique pair of Public and Private keys from the RTA and store it inside the vehicle object. The RTA also keeps a record of the Public Private key pair and stores it alongside the vehicle ID in its database. Whenever a vehicle enters a new RTA zone it assigns a new pair of Public Private keys to it.

c. Check for Connection with RSU and verify security of the communication.

This is the step for R2V verification where the RSU broadcasts its details to all nearby vehicles in its range and shares its Public Key with them. Also then the identity of the RSU is verified.

3. Scenario.py

This file provides the experimental settings on the vehicle speed distribution and the inter-vehicle distance distribution that can reflect different traffic situations. The scenario class defines some basic simulation components needed to initialize and update the overall vehicular network state.

a. Import class from Vehicle.py and RSAfucntions.py

```
from vehicle import Vehicle, Mobility
from RSAfunctions import *
```

b. Store vehicleID in *Vehicle.csv*

```
    with open("vehicleList.csv", 'a', newline='') as f:
        writer = csv.writer(f)
```

c. RSU sends RSU keys to all vehicles

RSU broadcasts its keys to all vehicles in a given range and shares its Public key and RSU ID with them.

```
    # step 2 : RSU -> *
    veh1.receiveMessage()
```

d. Vehicle and RSU verification begins

This is the step of V2R verification , here a vehicle after receiving the RSU's public key it creates a Pseudonym and it is used to identify the vehicle in the network and also the vehicle verification is performed.

```
    print("Vehicle sending join request to RSU...")
    PSv = RSAencrypt(str(veh1.Vid), veh1.RsuPubKey)
    SIGv = RSAsign(str(PSv), veh1.vehicleKeys[1])
```

4. RTA.py

a. genenrateVehicleKeys: It generates vehicle key pair and store its value along with the corresponding Vehicle ID

```
def generateVehicleKeys(self, Vid):

    (publicKey, privateKey) = rsa.newkeys(1024)
    RTA.vehiclesInfo[Vid] = [publicKey,privateKey]
    return RTA.vehiclesInfo[Vid]
```

b. generateRsuKeys: It generates RSU key pair and store its value along with the corresponding RSU ID

```
def generateRsuKeys(self, Rid):

    (publicKey, privateKey) = rsa.newkeys(1024)

    RTA.rsuInfo[Rid] = [publicKey,privateKey]

    return RTA.rsuInfo[Rid]
```

c. RTA sends RSU and vehicle its key pair

```
rsuKeys =      rta.generateRsuKeys('1000')

vehicleKeys =  rta.generateVehicleKeys("4765")
```

d. Sends public key to RSU

For the verification of Vehicle to RSU connection. The RSU requests the Public Key of the vehicle corresponding to the Vehicle ID.

```
# step 4 : Vehicle ->

RSU PKv =

rta.receiveMessage()

rta.sendMessage(1000, PKv, "VehPKv")
```

5. *RSU.py*

a. Receives key pair from RTA

```
data = pickle.loads(data)

if(data[0] == 'keys'):

    self.rsuKeys = data[1]
```

b. Receives Join request from Vehicle

Here V2R connection request is received and Vid is decrypted .

```
    if(data[0] == 'joinRequest'):
        self.PSv = data[1][0]
        self.SIGv = data[1][1]

        Vid = RSAdecrypt(self.PSv, self.rsuKeys[1])
        # print("\n Vid is : ", Vid, "\n")
        return Vid
```

c. Requests Public key of vehicles from RTA for verification

The RSU requests the Public Key of vehicle from the RTA for the verification of identity of the sender .

```
    if(data[0] == 'VehPKv'):
        PKv = data[1]
        return RSAverify(str(self.PSv), self.SIGv,PKv)
```

d. If verification fails then add the vehicle to the blacklist.

Here the identity is verified from the Public key received from the RTA and if the verification fails then the vehicle's Pseudonym is added to the blacklist and any future connections from the same Pseudonym is automatically rejected.

```
print("Verifying vehicle with
RTA...") rsu1.sendMessage(5000,
Vid, "getPKv")
if(rsu1.receiveMessage()):
    print("vehicle
verified") else:
    print("vehicle verification failed")
    blacklist.append(Vid)
```

6. *demo.py*

This file provides a basic simulation animation demo for the application of this VANET simulation project.

7. *communication_terminal.py*

This file defines a basic class representing the vehicular communication terminal.

For Decentralized Road Event Voting:

1. **Home Page:** A new car who joins the VANET is shared the network's blockchain and ledger. The user can create an event, vote on the current event, check the ledger for all votes and verify them, or directly view the results.

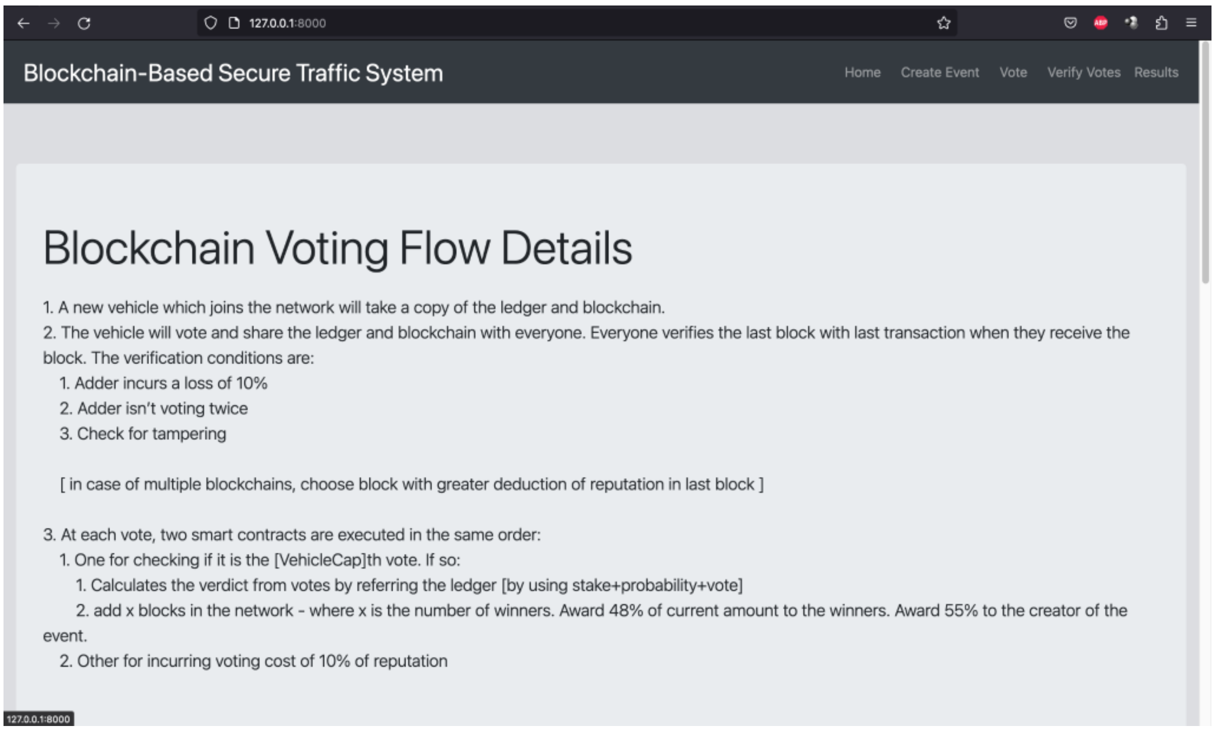


Figure 4: Home Page

2. **Create Event:** If an event is not already created, a user can create it by filling a form at Create Event webpage. The form requires the eventname, and the user's private key to verify them. This is the same private key assigned to the vehicle to them by the RTA. Once the user is verified by their public key, filling the form creates two cases of the event automatically:
    a. 'EventName' has occurred
    b. 'EventName' has not occurred

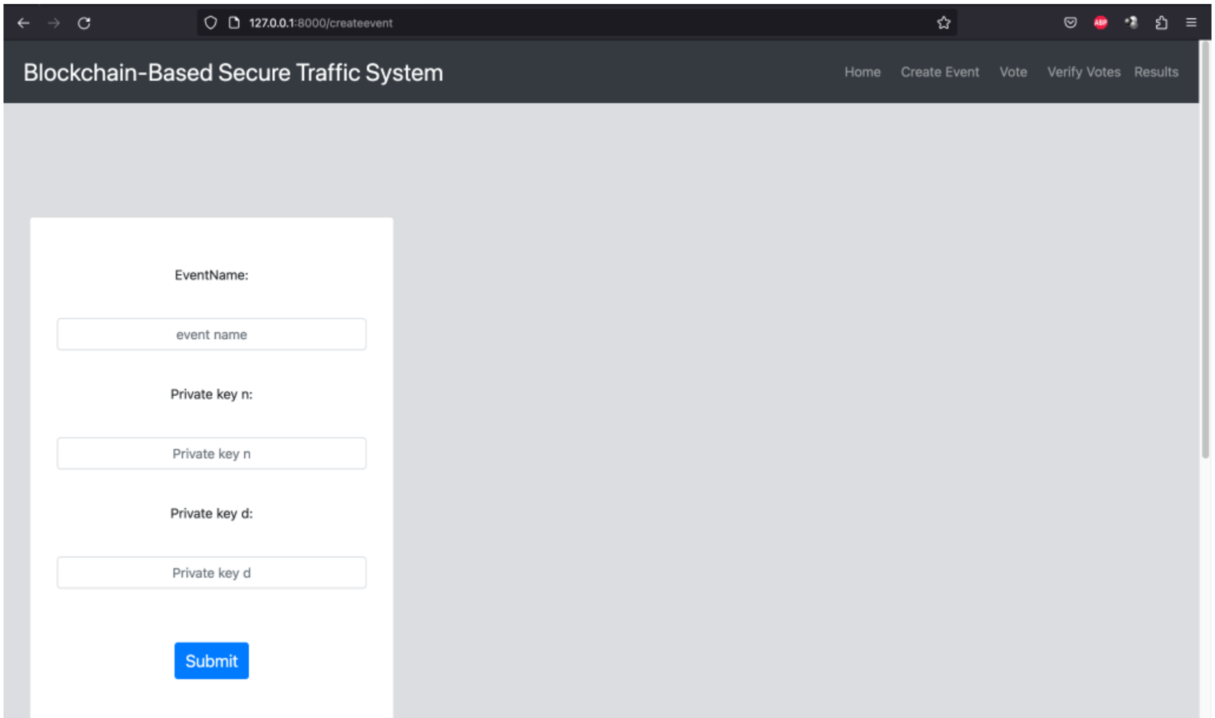


Figure 5: Event Creation

3. **Vote:** If an event is created, a user can vote via the Vote page. The user has to enter their private key to vote. This key is used to verify the voter, whereas the public key acts as the addresses of the user that they use for transactions. Here, instead of some currency like Ethereum or Bitcoin, *reputation* is used. Whenever a user votes, they put 10% of their reputation at stake, which is deducted from their 'account'. This 'transaction' is stored in the blockchain and public ledger. In each transaction, following details are stored:
    a. - Event name
    b. - Location and time
    c. - Creator's public key
    d. - Voter's public key
    e. - Voter's current reputation

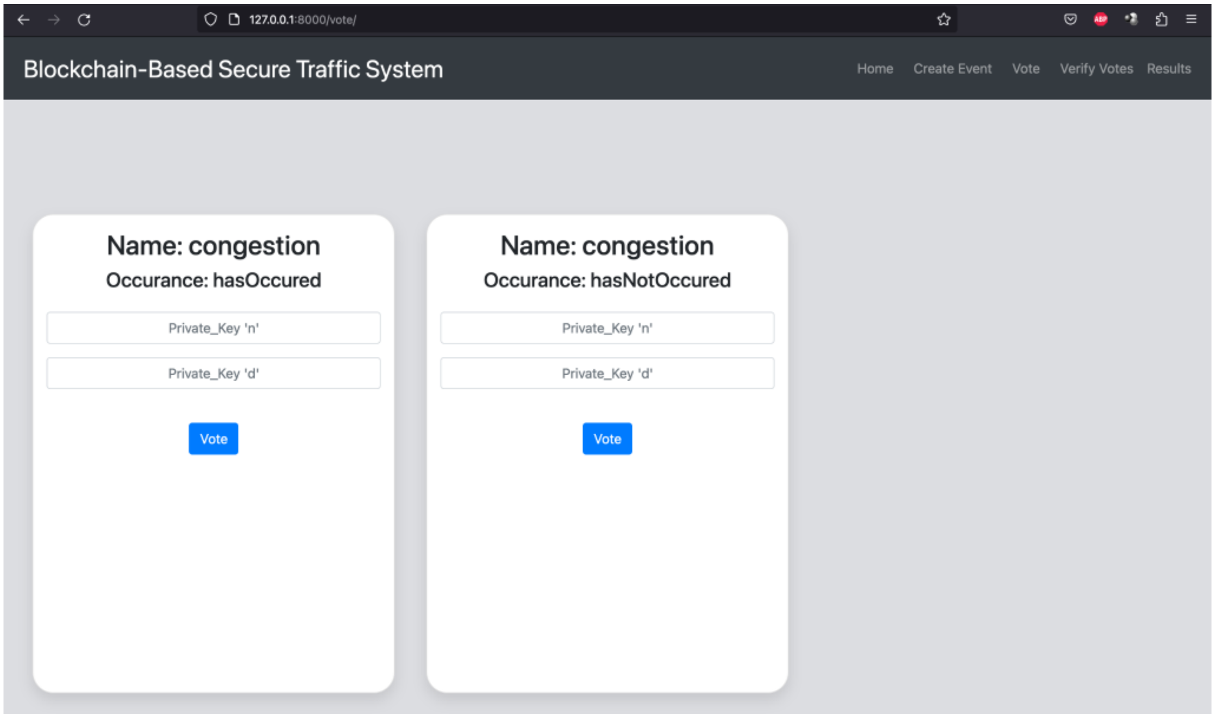


Figure 6: Voting of event

After each vote, two smart contracts are executed:

a. *SmartContractToWithdraw*: used for incurring voting cost of 10% of reputation.
b. *SmartContractToAward*: used to check if it is the [VehicleCap]th vote. If so:
   i. Calculate the verdict from votes by referring to the ledger. Verdict is calculated using a randomization based algorithm
   ii. Add x transactions in the ledger (and blockchain) - where x is the number of winners. Award 48% of current amount to the winners. Award 55% to the creator of the event.

Randomization Based Algorithm: The given code is a voting system that utilizes reputation and randomness to select voters for a poll. It first checks whether the number of votes exceeds a predefined limit, and if so, retrieves a list of all votes and their respective voters from the ledger. It then calculates the total reputation of all the voters and randomly selects a set of voters. After selecting voters, the algorithm increases the count of the event that was voted for by the selected voters and determines the verdict based on the event with the highest count of votes. It adjusts the reputation of voters based on their voting behavior, and saves the updated reputation of each voter in the database. The algorithm ensures the fairness and integrity of the voting process.

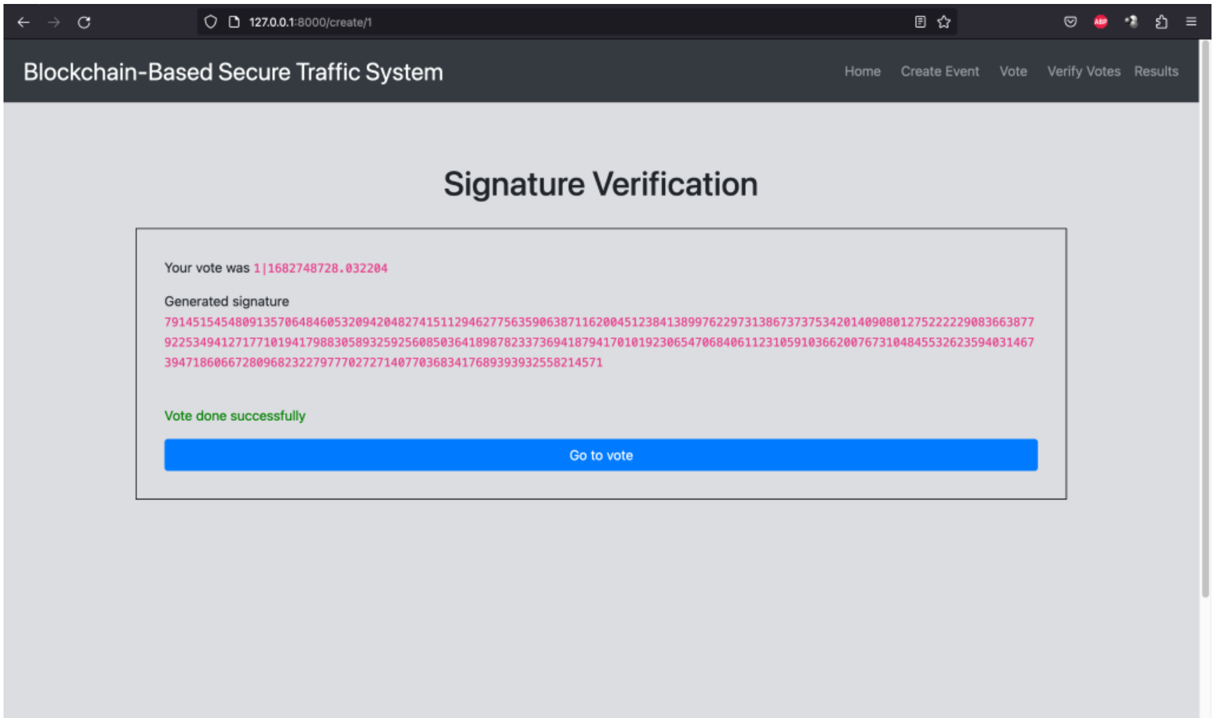


Figure 7: Successful voting

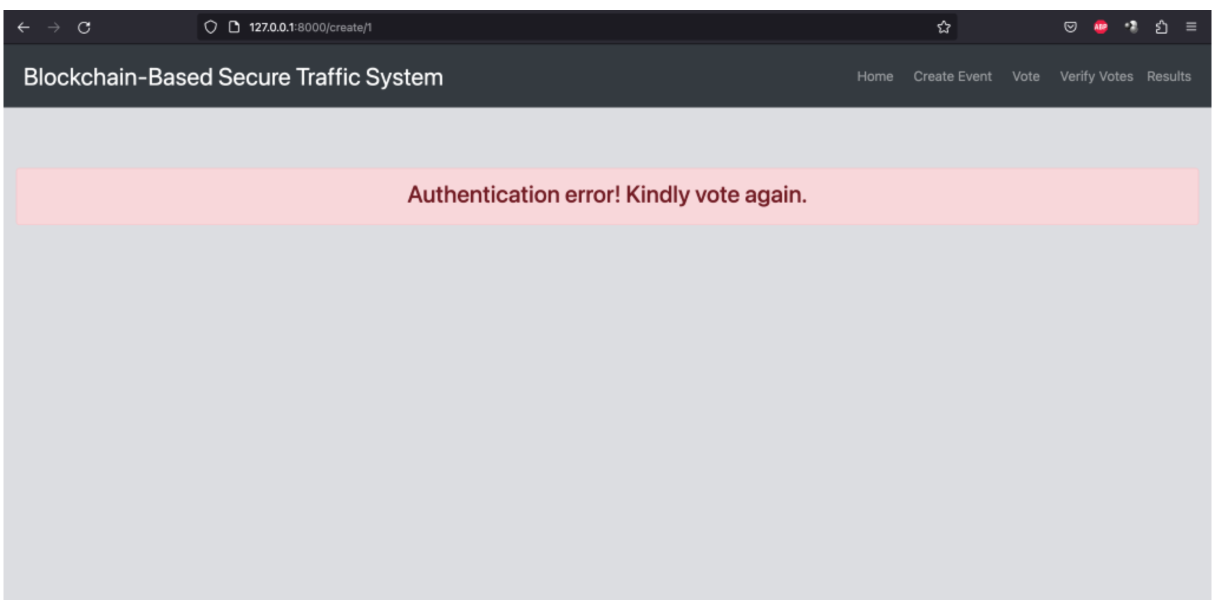


Figure 8: Unsuccessful voting: wrong private keys entered

4. **Verify Votes**: This page is used to view the ledger. Before loading the ledger, the page verifies the integrity of the blockchain by checking that all blocks are linked together and that none of the transactions have been tampered with. There are three verifications done

when the webpage is opened:

a. *verifyTampering*: this function verifies the integrity of each block in the blockchain by comparing the Merkle root hash of the block to the calculated Merkle root hash of the transactions stored in that block.

b. *verifyStake*: this function verifies that the user has indeed put 10% of their reputation at stake when voting. A malicious user may modify their code to incur smaller losses and larger gains. Other users will verify this and reject the blockchain..

c. *verifyDoubleVoting*: This function detects cases of double voting by checking whether a user has cast multiple votes for the same event. In a hypothetical scenario where a user has honestly voted twice for the same event, they will receive credit for both votes. This creates an opportunity for malicious users with low reputations to quickly boost their standing by casting multiple honest votes for a single event.

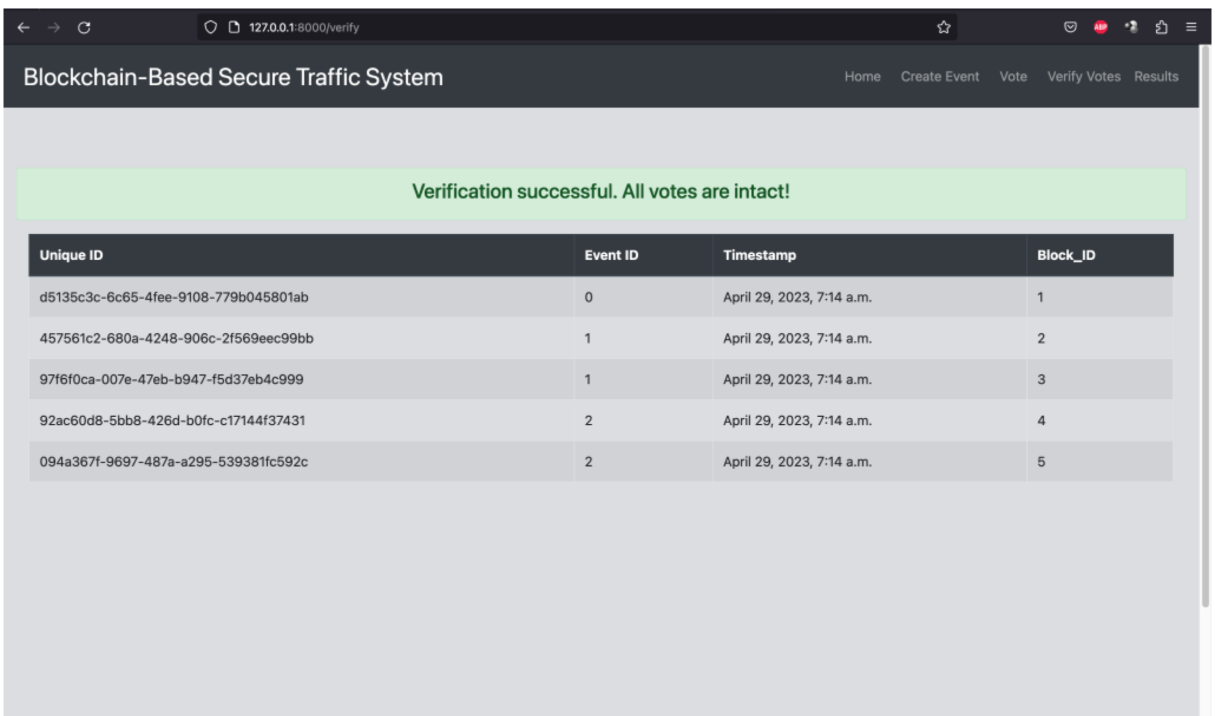


Figure 9: Verify Voting screen - ledger

5. **result:** Calculates the results of the election by counting the number of votes for each candidate and displaying them in descending order of popularity.

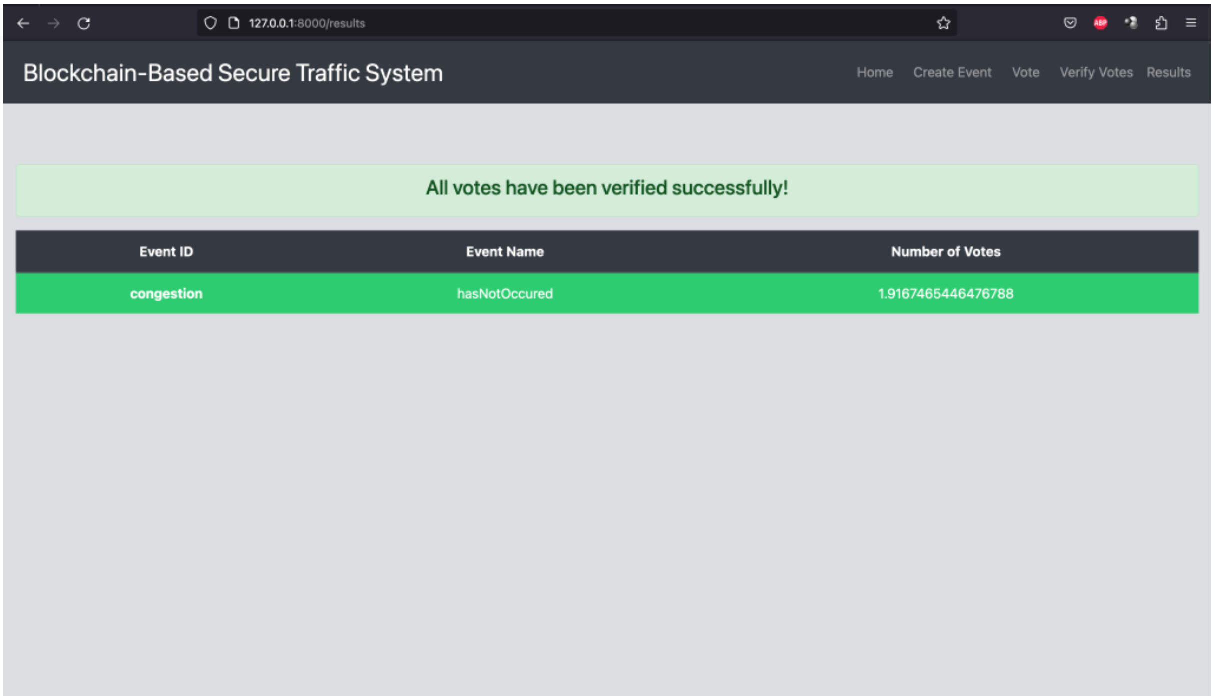


Figure 10: Result page

Other essential sections from the code:

1. generateBlock: Generates a block by adding a nonce and calculating the hash of the previous block, the current block's transactions, the current block's timestamp, and the current block's nonce. If the hash satisfies the requirements, the block is added to the blockchain and saved into the sqlite database.

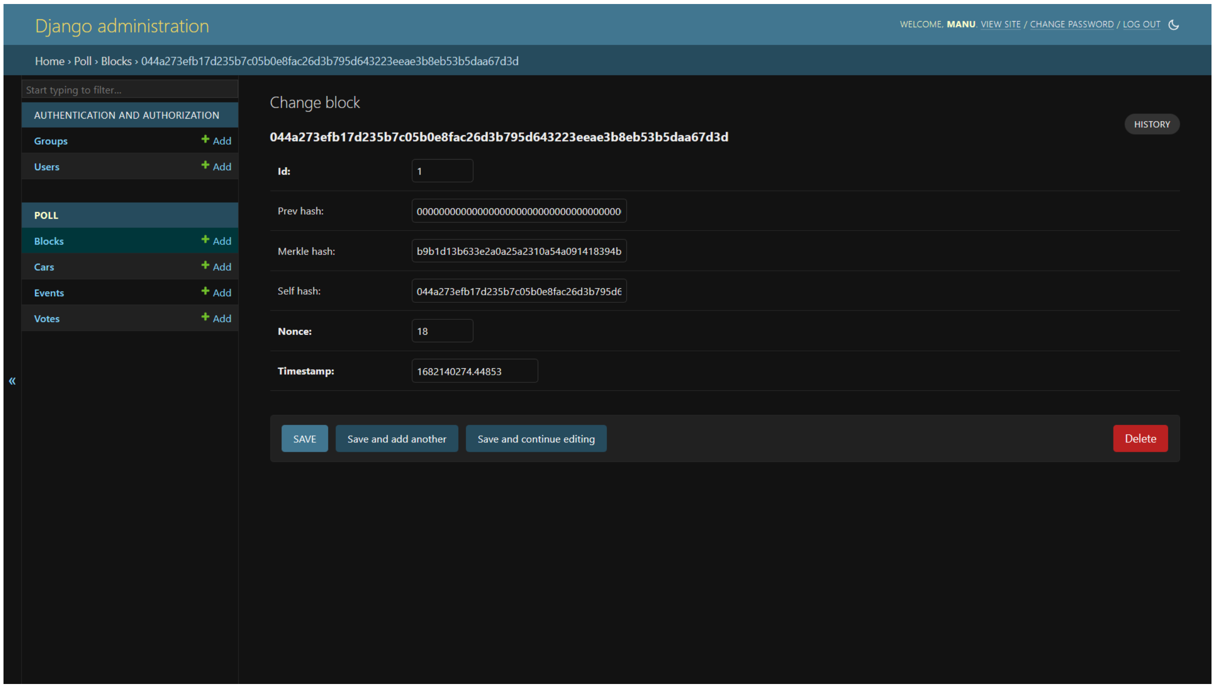


Figure 11: Block in Blockchain

2. merkleTree.py

The voter is verified through the use of a Merkle Tree.

Merkle Trees are commonly used in blockchain technology to ensure the integrity of data. In the context of e-voting, the Merkle Tree is used to verify the votes recorded in each block of the blockchain.

The code first retrieves the total number of blocks in the blockchain using models.Block.objects.count(). It then iterates through each block using a for loop, retrieving the corresponding votes for each block using models.Vote.objects.filter(block_id=i).

Next, the code converts each transaction object into a string using [str(x) for x in transactions]. This is because Merkle Trees operate on strings.

A new instance of a Merkle Tree is created using merkleTree.merkleTree(), and the array of vote strings is passed to the makeTreeFromArray() function to create the Merkle Tree. The calculateMerkleRoot() function is then called to calculate the Merkle Root of the tree.

The code then checks if the calculated Merkle Root matches the Merkle Hash stored in the block. If they match, the code continues to the next block. If they do not match, it indicates that the votes recorded in the block have been tampered with. The block ID is added to the tampered_block_list and the loop continues to the next block.

Finally, the function returns the list of block IDs that have been tampered with.

Overall, this code verifies the authenticity of the votes recorded in each block of the blockchain by checking if the Merkle Root of the recorded votes matches the Merkle Hash stored in the corresponding block.

3. resources.py

1. *vehicleCap*: it is the benchmark no. of vehicles needed for consensus to declare the result of poll.
2. *minTimeForVoting*: it indicates the minimum time (in epoch format) until the voter can vote again at the same RSU.

4.3 Risk Analysis and Mitigation

Risk Id	Classification	Description of the risk	Risk Area	Probability	Impact	RE (P * I)
1	Environment	RSUs may not be reliable in extreme weather conditions	Integration and Test	0.3	0.4	0.12
2	Budget	Setting up RSUs and RTAs to cover every road would require huge investment from the government	Program Constraints (resources)	0.5	0.3	0.15
3	Performance	Risk of delay overhead due to poor broadband	Product Engineering	0.4	0.9	0.36

Table 4:Shows Risk Area

Description of the risk	RE (P * I)	Mitigation method
RSUs may not be reliable in extreme weather conditions	0.12	Use the same mitigation methods used by FASTag to withstand extreme weather conditions
Setting up RSUs and RTAs to cover every road would require huge investment from the government	0.15	With time, development in advanced vehicles with VANET communication abilities will come to the road. The framework will be introduced first in metro cities, and then to smaller cities
Risk of delay overhead due to poor broadband	0.36	Backup wi-fi network infrastructure, with 5G broadband connection

Table 5:Description of risk with solution for it

CHAPTER 5

TESTING

5.1 Testing Plan

The testing plan includes simulating different scenarios of traffic to compare the ratio of message carriers vs time and propagation distance vs time graphs. Other types of tests to be performed are listed in the table in the next subsection, to check for the correctness, integration and security of the program as a whole and how it performs.

Software items: ● Windows 10 OS ● Visual Studio Code ● Powershell
Hardware items: ● 2 GB (64-bit) RAM ● Processors of Intel i3 or higher ● NIC card for socket programming

5.2 Component decomposition and type of testing required

The main components of the project are:

1. **Simulation**: The simulation uses demo.py, scenario.py, Vehicle.py and communication_terminal.py

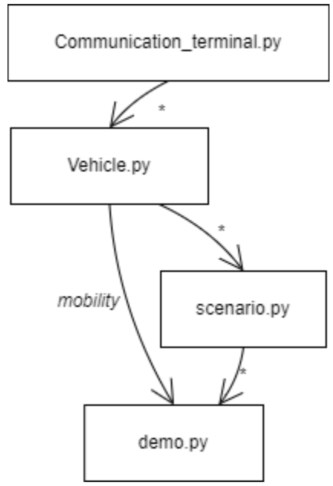


Figure 12: Main components for simulation

2. **Vehicle authentication framework**: The authentication framework uses Vehicle.py, RSU.py and RTA.py.

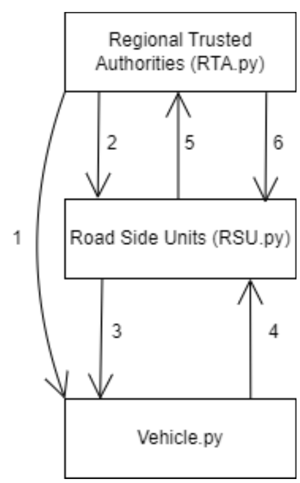


Figure 13: Flow of data transferred among different files

3. **Decentralized Road Event Voting**: The main files involved are views.py, models.py and merkleTree.py.

Type of Test	Comments/Explanations	Software Component
Requirements testing	Done to clarify whether project requirements are feasible or not in terms of time, resources and budget.	Overall code
Unit and functional Testing	Individual units of source code are tested to determine whether they are fit for use. In functional testing, each function is tested by giving the value, determining the output, and verifying the actual output with the expected value.	Individual python code files
Integration Testing	units or individual components of the software are tested in a group in order to expose defects at the time of interaction between integrated components or units.	Simulation's integration with vehicle authentication framework
Security Testing	Security testing is an integral part of software testing, which is used to discover the weaknesses, risks, or threats in the software application	Vehicle authentication framework; Decentralized Road Event Voting

Table 6: Testing done by individual tests

5.3 List of all Test Cases

5.3.1 Requirements testing

We check requirements by measuring the memory used by the project. We will use the Tracemalloc module in Python to measure the peak size of block traced in bytes:

```
print("Memory used by demo.py is:")
print(tracemalloc.get_traced_memory())
```

```
Memory used by demo.py is:
(4149037, 4169634)
```

Thus the program uses 4169634/1000000 = 4.17MB of memory to execute, which qualifies as a feasible amount of memory for even low end systems.

```
Block 2 has been mined
(1235410, 1571525)
```

Similarly, the program uses 1571525/1000000 = 1.57MB of memory to execute, which again qualifies as a feasible amount of memory for even low end systems.

5.3.2 Unit and Functional Testing

All main files (RTA.py, RSU.py and demo.py) successfully run individually, as exception handling was carefully done. The simulation ends when IndexError is faced by the compiler, as expected.

```
step 0: begins
--------------
Initializing...
Initialization done
step 0: ends

step 1: begins
--------------
RTA -> RSU: keys sent
step 1: ends
```

```
Step 2: RSA verified
```

```
step 2 & 3: begins
------------------
RSU -> *: broadcasting RSU details...
RSU -> *: broadcasted
step 2 & 3: ends
```

```
Step 4 : begins
---------------

Vehicle sending join request to RSU...
Request sent
step 4: ends
```

```
step 5: begins
--------------
Verifying vehicle with RTA...
vehicle verified
step 5: ends
```

For Decentralized Road Event Voting, all webpages are functioning successfully.

Blocks are being correctly mined and votes are being correctly casted:

```
[24/Apr/2023 04:34:21] "POST /create/0 HTTP/1.1" 200 2423
[24/Apr/2023 04:34:22] "POST /login/ HTTP/1.1" 200 2442
[24/Apr/2023 04:34:24] "POST /login/ HTTP/1.1" 302 0
[24/Apr/2023 04:34:24] "GET /vote/ HTTP/1.1" 200 3417
manu

casted ballot: 0|1682291069.718035
```

```
Block 2 has been mined
(1235410, 1571525)
False
[24/Apr/2023 04:34:21] "POST /create/0 HTTP/1.1" 200 2423
[24/Apr/2023 04:34:22] "POST /login/ HTTP/1.1" 200 2442
[24/Apr/2023 04:34:24] "POST /login/ HTTP/1.1" 302 0
[24/Apr/2023 04:34:24] "GET /vote/ HTTP/1.1" 200 3417
manu
```

5.3.3 Integration Testing

Vehicle.py acts as the main integration between simulation and authentication framework code. Its class contains methods for simulation purposes such as update_acceleration() and update_position(), and for authenticating itself through socket programming with sendMessage() and receiveMessage() methods.

The simulation runs successfully; before the appearance of each vehicle in the scene, it is verified by the RSU and RTA. Unverified vehicles are blacklisted and not included in the scene.

```
    if(rsu1.receiveMessage()):
        print("vehicle verified")
    else:
        print("vehicle verification failed")
        blacklist.append(Vid)
```

5.3.4 Security Testing

The main security concern is that private keys of RSUs and vehicles are not leaked or transmitted through an insecure medium. In theory, a vehicle will be provided its keys at the border of a region by the RTA (e.g. at toll bridges). During verification of the vehicle, the private keys are never transmitted to any entity. rsa library of python was used for encryption, decryption and digital signatures via RSAfunctions.py.

```
def RSAencrypt(pt, key):
    return rsa.encrypt(pt.encode('ascii'), key)

def RSAdecrypt(ct, key):
    try:
        return rsa.decrypt(ct, key).decode('ascii')
    except:
        return False
```

```
def RSAsign(message, key):
    return rsa.sign(message.encode('ascii'), key, 'SHA-1')

def RSAverify(message, signature,
    key): try:
        return rsa.verify(message.encode('ascii'), signature, key,) ==
    'SHA-1' except:
        return False
```

On tampering with the votes, verification function is correctly able to detect tamperin

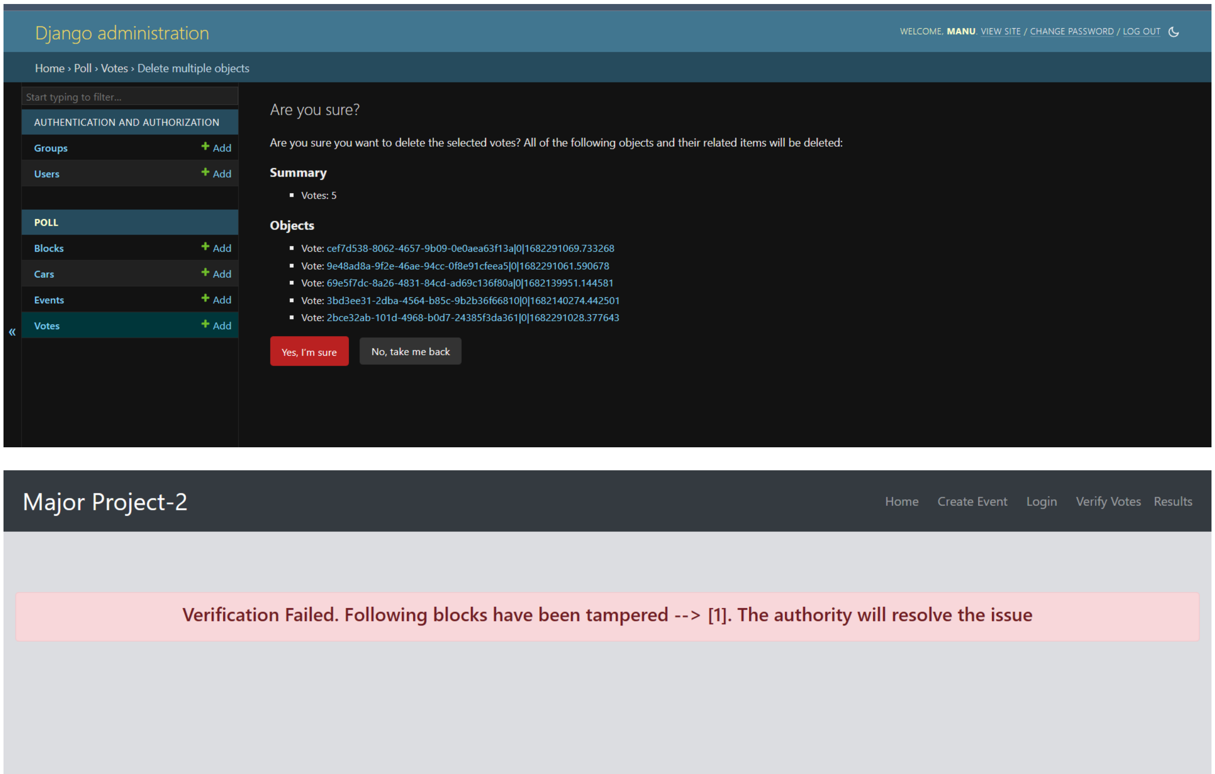


Figure 14: Tampering verified

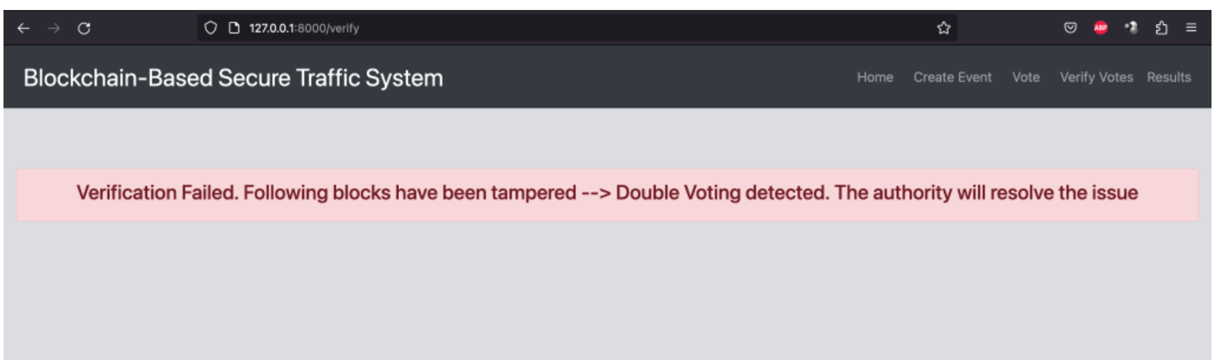


Figure 15: Double voting verified

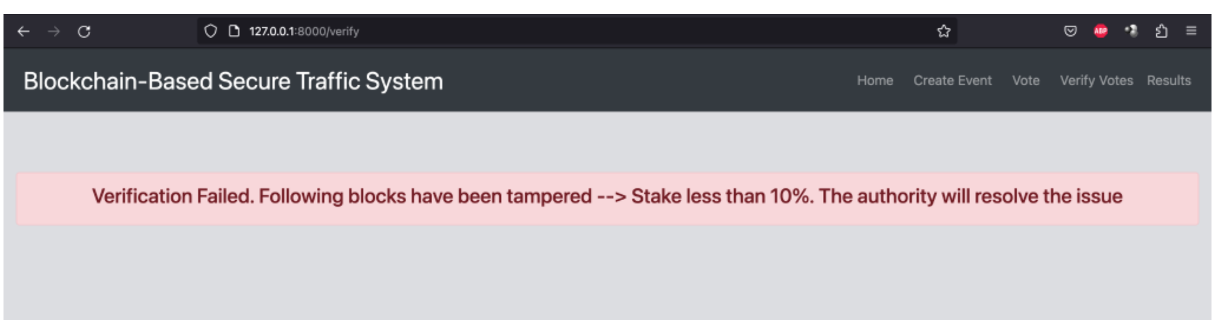


Figure 16: Stake verified

5.4 Error and Exception Handling

ConnectionRefusedError: The sender keeps trying to send data to the receiver until the receiver port is opened for receiving messages.

```
checker = 1
while (checker == 1):
    try:
        veh1.sendMessage(1000, [PSv, SIGv], 'joinRequest')
        checker = 0
    except ConnectionRefusedError:
        print("connecting...")
```

If the blockchain has been tampered with, it may be possible that no block may be returned when verifying votes. A try-catch case has been added for such cases:

```
def verifyVotes():
    block_count = models.Block.objects.count()
    tampered_block_list = []
    for i in range (1, block_count+1):
        try:
            block = models.Block.objects.get(id=i)
        except:
            return "tampered"
        transactions = models.Vote.objects.filter(block_id=i)
        str_transactions = [str(x) for x in transactions]
```

5.5 Limitations of the solution

1. The most prominent limitation is the lack of practical demonstration.
2. The framework is prone to DoS attacks by malicious attackers, who can easily disrupt the communication between legitimate vehicles and RSUs by flooding the channel.
3. Any delay overhead (such as due to poor broadband) will affect the efficiency and timely communication of vehicles.
4. Implementation is based on RSA, which is less secure and has smaller key size when compared with now popular Elliptic Curve Cryptography (ECC).

CHAPTER 6

RESULT, CONCLUSION AND FUTURE WORK

6.1 Results

Time analysis:

```
Time taken for generating and transmitting keys:
0.0029985904693603516
```

```
Start time of protocol is:
1670420416.7047136
```

```
End time of protocol is:
1670420416.729274
```

Thus, the total time of verifying the vehicle took 0.0246 seconds.

Plots for different mobility scenario:

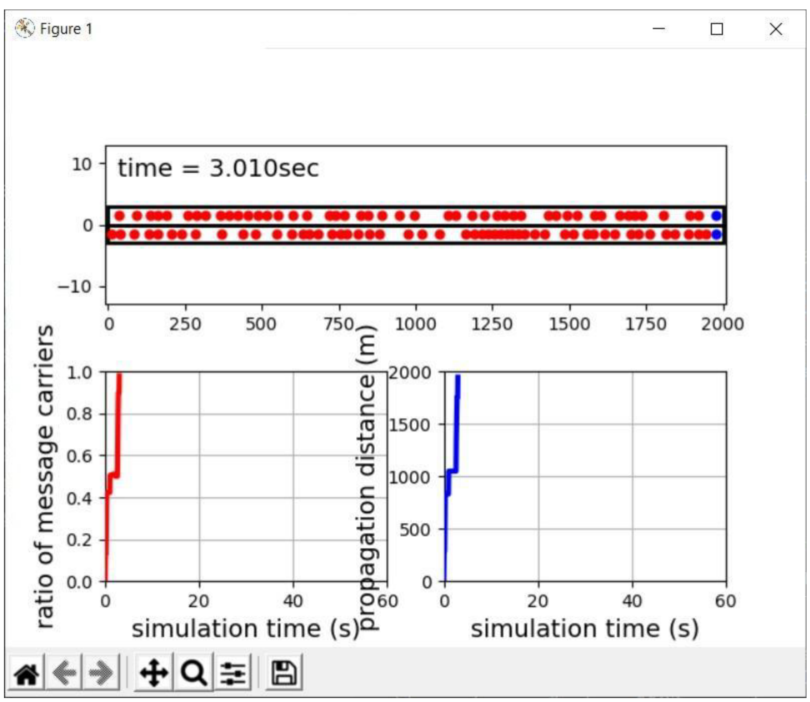


Figure 17. Illustration of Freeway in light traffic

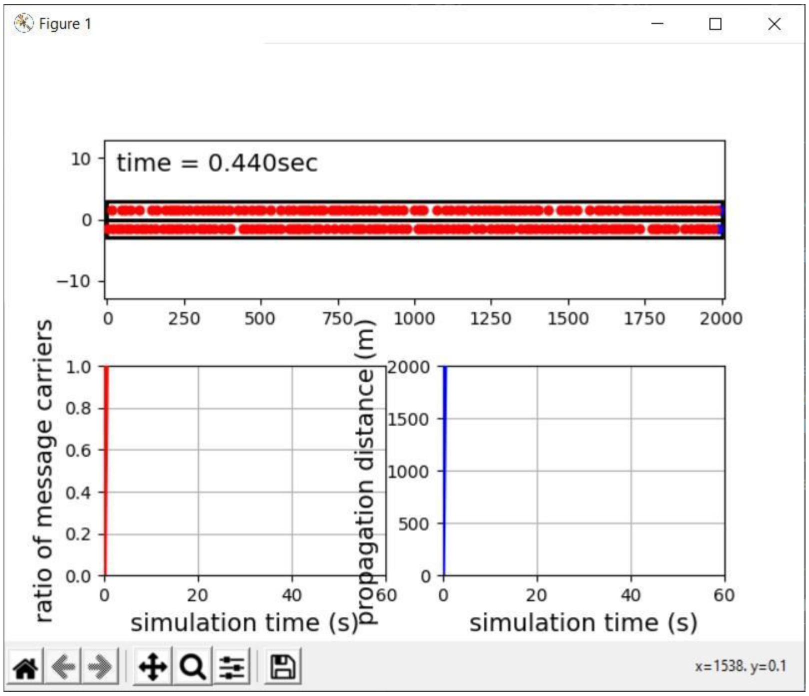
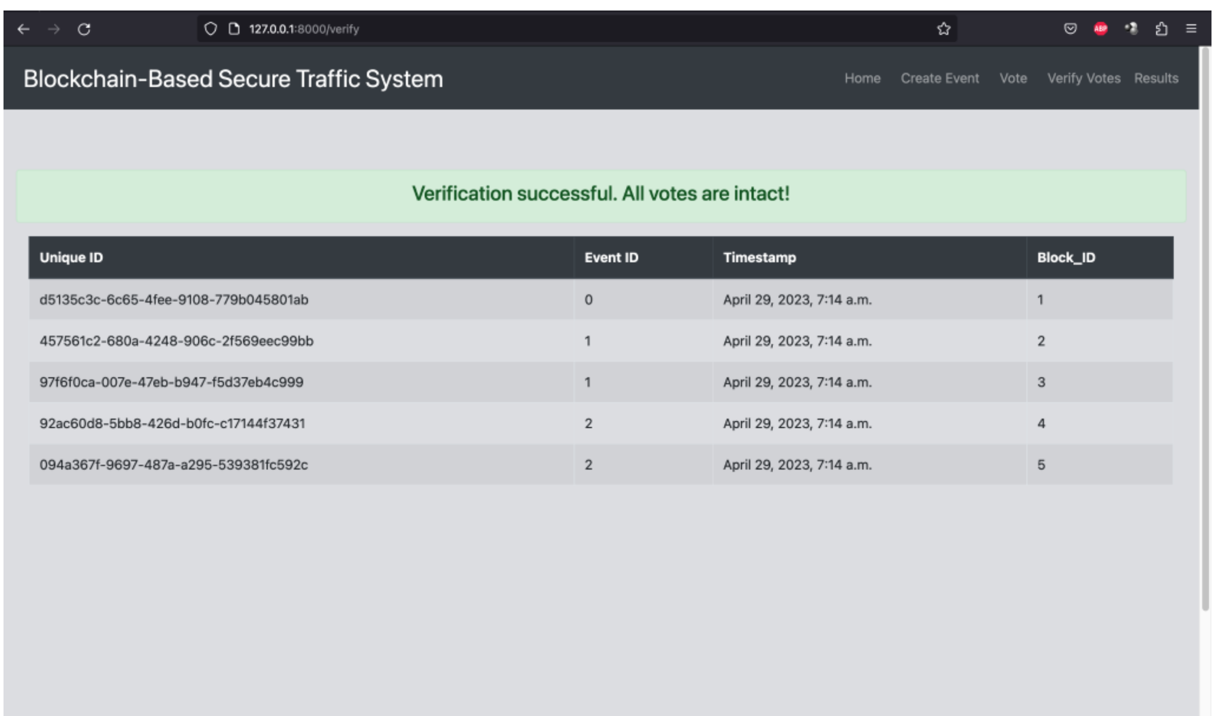


Figure 18. Illustration of Freeway in Rush Hour

Blocks being successfully validated and added in blockchain:



Unique ID	Event ID	Timestamp	Block_ID
d5135c3c-6c65-4fee-9108-779b045801ab	0	April 29, 2023, 7:14 a.m.	1
457561c2-680a-4248-906c-2f569eec99bb	1	April 29, 2023, 7:14 a.m.	2
97f6f0ca-007e-47eb-b947-f5d37eb4c999	1	April 29, 2023, 7:14 a.m.	3
92ac60d8-5bb8-426d-b0fc-c17144f37431	2	April 29, 2023, 7:14 a.m.	4
094a367f-9697-487a-a295-539381fc592c	2	April 29, 2023, 7:14 a.m.	5

Figure 19: Successful addition of Votes in ledger

Declaration of result

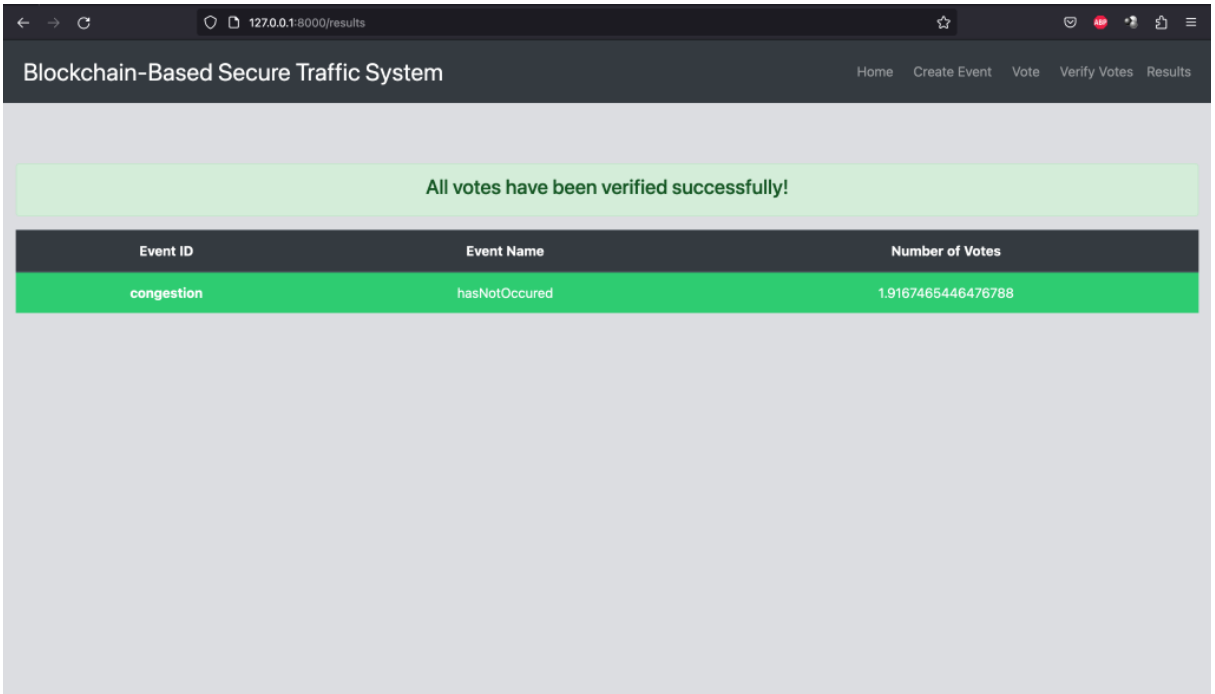


Figure 20: Result page

6.2 Conclusion

The total time of verifying the vehicle took 0.0246 seconds. Of course, in a real-life scenario, the time will largely differ due to latency. This would depend upon the bandwidth, network and location of RSU and RTA. But with this result, we may conclude that the protocol itself is efficient to be implemented in real-life.

It is also observed that in a freeway with light traffic, a message takes about 3 seconds to propagate 2000 meters in perfect scenario, where no latency delay is assumed. The time of propagation of a message is inversely proportional to the density of traffic.

Thus, a novel secure vehicle verification protocol has been proposed. Simulation shows that the proposed framework is feasible to the VANET environment for efficient V2V communication.

Through our project, we have demonstrated the feasibility of this approach and developed a proof-of-concept system that can be further refined and scaled for real-world applications. We have also identified some potential challenges and limitations, such as the need for wider adoption and education on blockchain technology.

Overall, decentralized road event voting holds great promise for improving the management of road events and ensuring that all stakeholders have a voice in the decision-making process. As such, we believe that further research and development in this area are warranted, and we look forward to seeing how this concept evolves in the years to come.

6.3 Future Work

1. Scalability: As smart traffic systems continue to grow and become more complex, the scalability of blockchain-based solutions needs to be improved to handle the increased traffic and data volume. Researchers can explore solutions such as sharding and off-chain transactions to address this issue.

2. Interoperability: Smart traffic systems often involve multiple stakeholders and systems, and ensuring interoperability between different blockchain-based solutions is critical. Future work can focus on developing interoperability standards and protocols to enable seamless communication between different blockchain networks.

3. Privacy: While blockchain technology provides a high level of security, privacy concerns still need to be addressed. Researchers can explore solutions such as zero-knowledge proofs and differential privacy to ensure that personal data is kept private while still allowing for secure communication and transactions.

4. Energy efficiency: The energy consumption of blockchain-based solutions is a significant concern, and researchers can explore ways to reduce the energy requirements of these systems while still maintaining the security and efficiency of the network.

5. Real-world implementation: Finally, future work can focus on implementing blockchain-based solutions in real-world smart traffic systems to test the feasibility and effectiveness of these solutions. This can involve collaboration between researchers and industry partners to develop and deploy these systems in a controlled environment.

CHAPTER 7
REFERENCES

[1] Alladi, Tejasvi & Chamola, Vinay & Sahu, Nishad & Venkatesh, Vishnu & Goyal, Adit & Guizani,Mohsen. (2022). A Comprehensive Survey on the Applications of Blockchain for Securing Vehicular Networks.

[2] Y. -T. Yang, L. -D. Chou, C. -W. Tseng, F. -H. Tseng and C. -C. Liu, "Blockchain-Based Traffic EventValidation and Trust Verification for VANETs," in IEEE Access, vol. 7, pp. 30868-30877, 2019, doi:10.1109/ACCESS.2019.2903202.

[3] Villacampa, J. A. (2019). Towards a blockchain-based private road traffic management implementation [Master's thesis, Luleå University of Technology]. http://www.diva-portal.se/smash/get/diva2:1321760/FULLTEXT01.pdf

[4] Pal, S., Hill, A., Rabehaja, T., & Hitchens, M. (2022). VeriBlock: A Blockchain-Based VerifiableTrust Management Architecture with Provable Interactions (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2206.05676

[5] E. -h. Diallo, O. Dib, N. R. Zema and K. Al Agha, "When Proof-of-Work (PoW) based blockchain meets VANET environments," 2021 12th International Conference on Information and Communication Systems (ICICS), 2021, pp. 336-343, doi: 10.1109/ICICS52457.2021.9464609.

[6] Saad, M.; Khan, M.K.; Ahmad, M.B. Blockchain-Enabled Vehicular Ad Hoc Networks: A Systematic Literature Review. Sustainability 2022, 14, 3919. https://doi.org/10.3390/su14073919

[7] Mikavica, B., Kostić-Ljubisavljević, A. Blockchain-based solutions for security, privacy, and trust management in vehicular networks: a survey. J Supercomput 77, 9520–9575 (2021). https://doi.org/10.1007/s11227-021-03659-x

[8] Leo Mendiboure[0000−0001−6643−9567] ,Blockchain and Cooperative Intelligent

Transport Systems: Challenges and Opportunities . http://dx.doi.org/10.1007/978-3-030-99004-6_4

[9] S. Maaroufi and S. Pierre, "BCOOL: A Novel Blockchain Congestion Control Architecture Using Dynamic Service Function Chaining and Machine Learning for Next Generation Vehicular Networks," in *IEEE Access*, vol. 9, pp. 53096-53122, 2021, doi: 10.1109/ACCESS.2021.3070023.

[10] Hesham El-Sayed , Henry Alexander, Parag Kulkarni, Manzoor Ahmed Khan,Rafidah Md Noor , and Zouheir Trabelsi . A Novel Multifaceted Trust Management Framework for Vehicular Networks. https://doi.org/10.1109/TITS.2022.3187788

[11] D. Manivannan , Shafika Showkat Moni , Sherali Zeadally. Secure authentication and privacy-preserving techniques in Vehicular Ad-hoc NETworks (VANETs) . https://doi.org/10.1016/j.vehcom.2020.100247

[12] JianshanZhou. Vehicular_AdHoc_Networks_Simulation https://github.com/JianshanZhou/Vehicular_AdHoc_Networks_Simulation

[13] Sunilkumar S. Manvia, Shrikant Tangadeb . A Survey on Authentication Schemes in VANETs for Secured Communication. https://doi.org/10.1016/j.vehcom.2017.02.001

[14] V.Lakshmi Praba, A.Rani Chitra ,Isolating Malicious Vehicles and Avoiding Collision between Vehicles in VANET. https://doi.org/10.1109/iccsp.2013.6577169

[15] Ikram Ali, Alzubair Hassan, Fagen Li . Authentication and privacy schemes for vehicular ad hoc networks (VANETs):https://doi.org/10.1016/j.vehcom.2019.02.002

[16] Jie Li, Huang Lu, Mohsen Guizani. ACPN: A Novel Authentication Framework with Conditional Privacy-Preservation and Non-Repudiation for VANETs. https://doi.org/10.1109/TPDS.2014.2308215

[17] M. Riley, K. Akkaya, and K. Fong, “A survey of authentication schemes for vehicular ad hoc networks,” Security Comm. Networks. pp. 1137–1152, 2011.

[18] R. A. Mollin, RSA and Public-Key Cryptography, Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2002.

[19] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,”

[20] J. Hoffstein, J. Pipher, and J. H. Silverman, “NTRU: A ring-based public key cryptosystem,” in Proc. ANTS-III, 1998, pp. 267–288.

[21] D. Boneh and M. Franklin, “Identity-Based Encryption from the Weil Pairing,” in Proc. CRYPTO, 2001, pp. 213–229.

[22] P. Barreto, H. Kim, B. Lynn et al., “Efficient Algorithms for Pairing-Based Cryptosystems,” in Proc. CRYPTO, 2002, pp. 354–369.

[23] A. Shamir, “Identity-Based Cryptosystems and Signature Schemes,” in Proc. CRYPTO, 1985, pp. 47–53.

[24] L.-A. Hîrţan, C. Dobre, and H. González-Vélez, “Blockchain-based Reputation for Intelligent Transportation Systems,” Sensors, vol. 20, no. 3, p. 791, Jan. 2020, doi: 10.3390/s20030791.