BlockV: A Blockchain Enabled Peer-Peer Ride Sharing Service

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Panchalika Pal

[Roll No: CS1701]

under the guidance of

Dr. Sushmita Ruj

Associate Professor
Cryptology and Security Research Unit

Indian Statistical Institute
Kolkata-700108, India

July 2019

To my parents

CERTIFICATE

This is to certify that the dissertation entitled **"BlockV: A Blockchain Enabled Peer-Peer Ride Sharing Service"** submitted by **Panchalika Pal** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by her under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Sushmita Ruj
Associate Professor,
Cryptology and Security Research Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would first like to express my sincere gratitude to my advisor, *Dr. Sushmita Ruj*, Cryptology and Security Research Unit, Indian Statistical Institute, Kolkata, for her continuous support, advice,enthusiasm, encouragement and motivation. Her guidance and knowledge helped me in pursuing good research and writing of this thesis. It is a privilege to have a supervisor who has constantly supported me, both academically and personally.

I would also like to thank *Laltu Sardar*, *Prabal Banerjee*, *Subhra Mazumdar*, *Nishant Nikram*, *Debendranath Das*, *Manish Kumar*, Cryptology and Security Research Unit, Indian Statistical Institute, Kolkata, who provided insight and valuable suggestions that greatly assisted the research.

I would like to express my deepest gratitude to the *Google* server for assisting me with every minute details I required for this research.

I take this opportunity to sincerely acknowledge the contributions of all the teachers of Indian Statistical Institute, who have deeply influenced my research acumen.

I thank my parents, for being my pillar of strength and supporting me in all my endeavours. Last but not the least, I want to thank my friends, for keeping me motivated at times when I felt like giving up.

Abstract

Today's ride sharing is a centralized trust based system where users trust the service providers for the ride set up, tracking, cancellation, fare calculation etc. Any malicious activity in the centralized server based system or a malicious driver or a malicious rider destroys the fairness involved in the ride and causes inconvenience to the parties. After the completion of the ride, the drivers are rated by the riders. There are possibilities that, a malicious rider can claim the refund with a fake complain and give the driver poor rating intentionally or a malicious driver follows a longer path unnecessarily and charges the rider more.

Current system is not capable of deciding the correctness of the objections raised by either parties regarding the ride and provides a biased outcome of each objections as per the centralized company's marketing strategies. In this context, we present BlockV, a blockchain enabled anonymous permissionless solution to ensure end to end fairness of the ride. The creation, completion, dissatisfaction or abortion of any ride will be written in the blockchain ledger, hence will be available to all participants in the peer to peer network. This simultaneously ensures the fairness in maintenance of the inbuilt reputation system. We have implemented a prototype in Ethereum private network and KOVAN test network and analyzed the security.

Keywords: *Permissionless Blockchain, Car sharing, Car riding, Route Fare Database, Driver, Rider, RSU, Anonymous, Elgamal, Reputation system, Fairness, Ethereum*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today's busy world demands ultra fast moving vehicles with the rapidly increasing growth in the vehicular technologies. Transportation shall be hassle free and trustworthy for every identity. The traffic system need to be highly time efficient. All these need converges to the speed of the vehicle which directly proportional to the congestion in the road. In studies it has been found that all over the world the number of vehicles per person has increased rapidly in last few years leading to slow vehicle movements due to restricted road capacity.

Thus, to avoid vehicle clogging in the roads, the primary goal is to encourage the full capacity utilization of the vehicles. In the other ways, the vehicle owner is encouraged to use its vehicle as the part of public transportation system and carry other persons as per the free capacity of the vehicle. This will lead to a decreased vehicle to person ratio and achieve a smooth traffic condition. The persons who are using their private car will switch to the car hiring methodologies, only if they find the procedures are simple, explainable, cost efficient and fair in all respect.

Here, we have presented BlockV, an architecture which follows these four principles and reflects a robust end to end solution. The procedure of hiring a car involves a negotiation for a fare against a ride chosen by the rider, a fair payment mechanism explainable to all and a decentralized system that ensures fair, trusted, cost effective transactions. The system which is being followed currently is the application based car sharing, where there exist many centralized servers monitoring every aspects of the ride mentioned above in different platforms of applications. It is hard to achieve fairness in the currently running systems and also there exists no common platform in this regard where the rider can go for a comparison and select the driver accordingly. Also, the fairness in the procedure is trusted to be implemented by the underlying service providers, which is not verifiable from the rider end. This kind of trust based system creates a point of dissatisfaction as the inner computations strategies are not clearly known to them. Hence, the car sharing to be widely appreciated among all the class of people, we have appreciated the role of decentralized peer to peer network of blockchain BlockV as the backbone of the architecture. The motivation behind

BlockV is firstly to ensure the $payment fairness$ where the breakup of the fare i,e. the cost of the ride for a particular path is computable by any peer of the network with the path details. Secondly, we present the $ride fairness$ where in the case of any dispute, addressed by the rider, the malicious driver or the malicious rider (in case of false allegation) will be penalized. BlockV collaborates with the Road Side Units (RSUs) to achieve fairness in this respect. We believe that our work will be useful to ride sharing services like Uber and Lyft.

1.1 Blockchain - Decentralized trust

A blockchain is a time-stamped series of immutable record of data that is managed by cluster of computers not owned by any single entity. Each of these blocks of data (i.e. block) are secured and bound to each other using cryptographic principles (i.e. chain). The blockchain network has no central authority. Since it is a shared and immutable ledger, the information in it is open for everyone to view. Hence, anything that is built on the blockchain is by its very nature transparent and everyone involved is accountable for their actions.

In order for a block to be added to the blockchain, however, four things must happen:

1. A transaction must occur.
2. That transaction must be verified.
3. That transaction must be stored in a block.
4. That block must be given a hash.

Blockchain technology accounts for the issues of security and trust in several ways. First, new blocks are always stored linearly and chronologically. They are always added to the "end" of the blockchain. Secondly, after that it is very difficult to go back and alter the contents of the block because each block contains its own hash, along with the hash of the block before it. Change in any content in any of the previous blocks, will change the hash values of all the later blocks. Thus one can easily detect the incident by looking at the last hash value. In order to change a single block, then, a hacker would need to change every single block after it on the blockchain. Recalculating all those hashes would take an enormous and improbable amount of computing power. In other words, once a block is added to the blockchain it becomes very difficult to edit and impossible to delete.

From the access point of view we have two kinds of blockchain- Permissioned (private) and Permission-less. Permissioned blockchains use an access control layer to govern who has access to the network where as in permission-less blockchain, or public, blockchain network the advantage is that guarding against bad actors is not required and no access control is needed. Another mix up category is also available- Hybrid Blockchain. It is a combination between different characteristics both public

and private blockchains have by design. It allows to determine what information stays private and what information is made public.

The analysis of public blockchains has become increasingly important with the popularity of bitcoin, Ethereum, litecoin and other cryptocurrencies. Blockchain-based smart contracts are proposed contracts that could be partially or fully executed or enforced without human interaction.

1.2 Problem Statement & Motivation

The inflation in the trend of using personal vehicles for every movement involved in day to day life has been created an alarming condition for traffic management systems worldwide. The increasing graph of the usage demands a high increasing graph of infrastructure and supports which will be saturated in near future. Thus it is very important to control and reduce the inflated trend of personal car usages. One man one car keeps the other seats unoccupied and unused increasing the traffic volume. To reduce the congestion, car sharing is only option. This will produce positive impact on vehicular volume utilization. Thus the problem of traffic congestion is reduced to the problem of car sharing.

The problem of car sharing involves the transparency in payment. Rider shall pay the requisite cost of the ride to the driver. There are certain risk involved int his payment. First, the driver can claim more money than actual cost. Hence, there should be an entity to monitor that. Secondly, the rider can pay less money than actual. This should also be monitored by some entity. Thirdly, there should be some system to decide what is the exact fare for a ride. Based on these criteria we have some centralized server based companies who monitor all these three requirements. But the problem of centralized servers are that they can easily be tampered. Hence, we can not trust a malicious server to restrict malicious activities.

Apart from the payment, rider can provide rating to the drivers as a token of satisfaction. A malicious rider can provide poor rating to a driver for not obeying to his illegitimate demand. The rider is protected in this respect as no penalization occurred for this mischievous activity. The driver may drop the rider at a wrong location or somewhere along the path. This is against the mutual agreement between the rider and the driver for the ride but if the rider is new in this route, it will create a great inconvenience to him. Till date all the dissatisfaction are reported to the centralized authority who can easily be manipulated. Also, for most of the cases we can never know the actions taken against the malicious user. Thus an user, willing to be the part of the a car sharing system, requires a decentralized trust based system which can be completely relied upon, open to all, handles dissatisfaction transparently. All users shall be equally treated. Any malicious activity if detected shall produce equal penalty for all users. This demands for the fairness in the ride.

Along these ride fairness and payment fairness we need user privacy as well. No one else other than the user shall be aware of the riding schedule, the frequency

of the ride, usual destination and find a link them all. This will break the user privacy. Hence, to motivate the common people for ride sharing we have to build up a system which thoroughly ensures *ride fairness*, *payment fairness* and *user privacy*. The decentralized objective motivates to search for a blockchain based solution as the blockchain provides the facilities of transaction validation by peer to peer network. Also the immutability of the transactions is ensured in this case. Any malicious activity can be easily noticed and the system to be malicious at least 51% of the peers in the network has to be malicious. This comes up with picture (Fig 1.1) of all drivers and riders connected in a peer to peer network and the rides are enlisted in blockchain ledger.

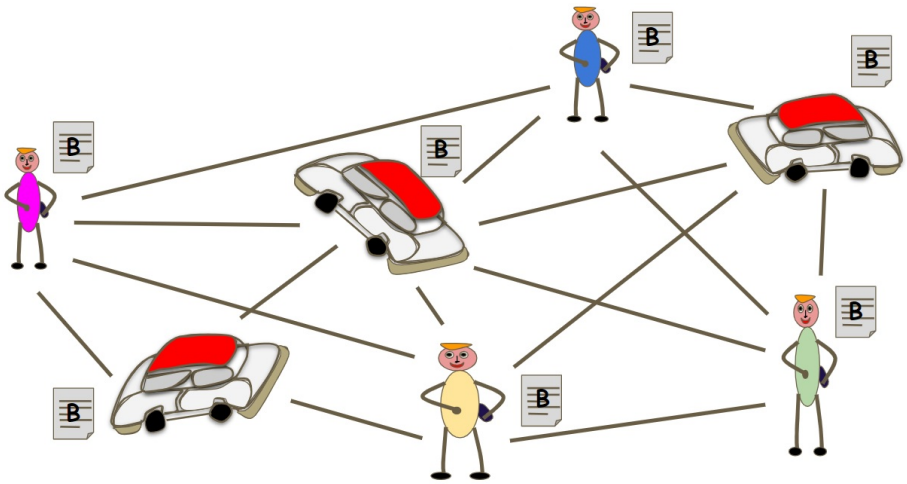


Figure 1.1: The system with decentralized trust

1.3 Our Contributions

In this thesis, we define and discuss in details a new blockchain based architecture for the car sharing system, named as BlockV [33]. With the motivations of simple robust cost effective reliable system design, we have introduced the fairness in payment calculation as well as in the ride. Any dispute regarding the failure in following the redefined and contracted path, if arises, will be solved by the peer to peer network as explained in later sections. The false allegations are also checked in the procedure to achieve the complete fairness from the users' perspective. By using off-chain transactions [10] and opportunistic arguments, we have substantially improved the performance while settling disputes. The user privacy is also addressed with this architecture. The security aspects of the architecture are also addressed

with this. Following that, we have implemented a prototype in Ethereum Virtual Machine private network and the results on gas costs, scalability, time effectiveness are included. The same contract is also deployed in the KOVAN testnet.

1.4 Organization of the thesis

The rest of the thesis is organized as follows: In Chapter 2, we have discussed related works and some prerequisites for BlockV . Chapter 3 describes the terminologies used in the next sections and basic definitions related to the scheme are mentioned. The high level views with the detailed procedures are also presented in this chapter. In the Chapter 4, we have addressed our security claims with the motivation of fairness in the system. We have presented the experimental results and analysis in Chapter 5. Next we have put the limitations and future scope of improvisation of this architecture with the conclusion in Chapter 6.

Chapter 2

Related Works

In this chapter we will discuss the related works and the public blockchain platform ethereum which has been used for implementation of BlockV.

2.1 Blockchains

There are several applications of blockchain currently being focused on. The most basic and simplest kind is the application in finance sector where decentralized crypto-currency is the main attraction. One of the famous crypto-currency is Bitcoin which has come up with the renaissance in the domain finance. The publication of the Bitcoin whitepaper [29] in 2008 was the first globally appreciated step taken in blockchain. Apart from Bitcoin we have now Litecoin [15], Namecoin [18], Peercoin [19], Dogecoin [4], where we use electronically coded addresses to make transactions. Traditional system requires a mediator like banker or a remittance company to ensure the trust in a transaction. With these crypto-currencies we have another application in finance-Asset management [3] with improved security, improved operational efficiency, client on boarding, stream lining portfolio management, clearances and settling of trades.

Blockchain can also be a suitable technology for Insurance companies [5]. Blockchain technology has been rather crucial in supporting various applications like supply-chain management [20], healthcare [27], data sharing for medical in cloud environments [51] and personal data [52], Internet of Things (IoT) with the optimized blockchain [7] to name just a few. It has revolutionized the sharing economy with applications like file sharing using Filecoin [35], music [30] and other digital content [48], digital certificates [47] etc. The basic purpose of using the blockchain is to shift the trust from centralized service providers to peers who maintain the blockchain. As long as majority of the peers are honest, the system is secure [23]. The fairness in the system achieved by the blockchain is shown in [9]. The Industrial Internet of Things are also benefited by the blockchain technology. The agricultural sector along with food supply chain management are also using blockchain technology [45]. Blockchain is

also used as a toll in Identity management [17].

There has been a tremendous amount of work in smart cities [1]. The blockchain based sharing services [44] can be included as the elements of the smart cities. The digital identities with blockchain [37] can also contribute to the smartness. Blockchain based hybrid networks [41] are well appreciated in the domain of smart cities. Thus a blockchain based distributive vehicular network is mentioned in [40]. The application of blockchain in vehicular technology was introduced in the field on Vehicle to Vehicle (V2V) communications [38], [42], [24]. The vehicle to grid communications through mobile IP is also depicted in [11]. The survey on existing authentication and privacy preserving schemes are presented in [14]. In addition to that we have cloud assisted video reporting techniques for 5G networks as describes in [28]. The lightweight model of trust in VANET(Vehicular Adhoc NETwork)s are well described in [25] and all of these come under the purview of Intelligent Transport Systems [12]. Dorri *et al.* [8] proposed a distributed solution to automotive security and privacy. Road Side Units (RSU) are assumed to be present which are capable of interacting with the vehicles and pass the messages according to the RSU communication protocols. This kind of architecture is described in [32]. A self managed vehicular network has been proposed in [21]. Blockchain enabled vehicular announcement has been proposed in [22]. This incentivises users to communicate traffic information while maintaining user privacy. An auto-insurance framework has been proposed in [31]. An anonymous authentication protocol was described [39]. The attacks in Industrial Control system as discussed in [13] is applicable to the vehicular industries as well. Hence a decentralized platform like the blockchain solution is considered to be one of the best available solutions.

In this thesis, we have specifically considered the application of blockchain in the domain of car sharing which was mentioned in [8] as one of the key blockchain applications that will come up in future. To increase the efficiency of the architecture, we have introduced the use of off-chain [34] as a part of the scheme. We have considered an secured inbuilt reputation system [36] with the representation of reputation score of each participating candidate in BlockV. Our experiments are performed in Ethereum Virtual machine (EVM) which was introduced in [46].

2.2 Ethereum

Ethereum [50] is an open programmable blockchain platform that allows anyone to build and use decentralized applications that run on blockchain technology. It is an open-source project built by many people around the world. Ethereum is designed to be adaptable and flexible. It is easy to create new applications on the Ethereum platform.

Like any blockchain, Ethereum also includes a peer-to-peer network protocol. The Ethereum blockchain database is maintained and updated by many nodes connected to the network. Each and every node of the network runs the EVM and executes the

same instructions. There are two types of accounts:

1. Externally Owned Accounts (EOAs), which are controlled by private keys
2. Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA

Users must pay a little amount transaction fees to the network. This helps the Ethereum blockchain to be protected from malicious computational tasks, for example, DoS attacks or infinite loops. The sender of a transaction must pay the fees in amounts of Ethereum's native value-token, ether.

The transaction fees are collected by the nodes or peers that receive, execute , verify and propagate transactions. The miners then group the transactions in blocks, and compete with one another for their block to be the next one to be added in to the blockchain. Miners are rewarded with ether for each successful block they mine. This provides the economic incentive for people to dedicate hardware and electricity to the Ethereum network.

The Ethereum Virtual Machines [16] can run smart contracts, written in solidity [6]. A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties.

Smart contracts are equivalent to the classes in object oriented programming. When we deploy the smart contract in Ethereum, we get an address analogous to the return value of new object() in Object Oriented Programming. We can write codes to interact with specific instances of the smart contracts. The *read* methods are quick but *write* methods takes sometimes because a write method requires a change in state variables of the Ethereum virtual Machines which creates new transaction and we need to wait for the next block to be mined.

Ethereum main network ("*mainnet*") deals with real money. Hence, before deploying a contract in main network with real assets we like to test the contract. Hence we deploy this to an Ethereum Test Network ("*testnet*"), which simulates Ethereum. Ether and tokens on a testnet are easy to obtain, and carry no real-world value. There are three popular public testnets, namely Ropsten, Kovan and Rinkeby. Ropsten is a proof-of-work blockchain that most closely resembles Ethereum. One can mine Ethers in this testnet. Kovan and RInkeby are proof-of authority blockchain started by parity team and geth team respectively. Here, Ether Can not be mined but has to be requested. Ethereum private test network can be built using geth and homebrew.

2.2.1 Overview of Ethereum Virtual machine

The Ethereum Virtual Machine(EVM) is a quasi-Turing complete machine which implements the execution model of Ethereum blockchain by providing a run-time

environment for smart contracts. The algorithm is implemented by smart contracts and the memory is represented by a virtual byte-array ROM. Computations are bounded by a parameter called gas. Before a smart contract is executed in ethereum, the gas cost is computed for each operation and paid when those are being executed. Some operations can not be executed if the program has less gas limit specified than required. With each new transaction, executed and mined in ethereum, the blockchain moves into a new state.

2.2.2 Architecture of EVM

The EVM is a simple stack-based architecture. Computation on the EVM is done using a stack-based bytecode language. The word size of the machine is 256-bits (32-byte), this is also the size of a stack item. The size of every item on the EVM stack is 256 bits. If the size of the data item is less than 256bits, it is padded with leading zeros. The stack has a maximum size of 1024. The memory model of the EVM is a simple word-addressed byte array. This means that the memory is an array of bytes, each byte is assigned its own memory address. The EVM has a storage model which is a word-addressable word array. Unlike the memory which is volatile, storage is non-volatile and it is maintained as part of the system state. All locations in both storage and memory are well-defined initially as zero.

2.2.3 Securing the Ethereum Blockchain with the EVM

The EVM imposes the following set of restrictions to secure the state of the system:

- Every computational step taken in process of executing a program must be paid for upfront, thereby preventing Denial-of-Service (DoS) attacks.
- Programs may only interact with each other by transmitting a single arbitrary-length byte array but they do not have access to each other's state.
- An EVM program can only access and modify its own internal state and may trigger the execution of other EVM programs, but not allowed to do anything else.
- Program execution is completely deterministic and produces identical state transitions for any similar type of implementation starting from an identical state.

These restrictions have helped to figure out the design decisions of the Ethereum state transition machine.

2.3 Digital Signature Algorithm

The Digital Signature Algorithm (DSA) [49] is a Federal Information Processing Standard for digital signatures. The scheme is based on the mathematical concept of modular exponentiation and the discrete logarithm problem. DSA is a variant of the famous Schnorr and Elgamal signature schemes.

2.3.1 Key Generation

There are two separate phases for key generation. One is generation of shared algorithmic parameters and the second one is typical for an user.

Shared Parameter Generation

- Choose an approved hash function $\mathcal{H}$ with output length $|\mathcal{H}|$ bits. If $|\mathcal{H}|$ is greater than the modulus length N, only the leftmost N bits of the hash output are used.
- Choose a key length L.
- Choose the modulus length N such that $N < L$ and $N \leq |\mathcal{H}|$
- Choose an N bit prime q and L prime p such that $p - 1$ is a multiple of q.
- Choose an integer h randomly where $2 \leq h \leq p - 2$.
- Compute $g = h^{\frac{p-1}{q}} \ mod \ p$. If g is 1 try with different h.

The algorithm parameters are (p, q, g).

User Key Generation

Given the set of shared parameters as generated using the steps described in the above procedure, this phase computes the key pair for a single user. The same procedure will be user by all the users to generate their respective public and private keys.

- Choose integer x randomly such that $1 \leq x \leq q - 1$
- Compute $y = g^x \ mod \ p$.

x is the private key and y is the public key. The user should publish only the public key and keep the private key secret.

2.3.2 Signing Algorithm

The user shall sign the message m as follows:

- Choose an integer k such that $1 \leq k \leq q-1$.
- Compute $r = (g^k \ mod \ p) mod \ q$. If r is zero, start with different random k.
- Compute $s = (k^{-1}(\mathcal{H}(m) + xr)) mod \ q$. If s equals zero, start with different random k.

The signature is (r, s).

2.3.3 Verification Algorithm

One can verify the signature (r, s) is a valid signature for message m as follows:

- Verify that $0 < r, s < q$.
- Compute $w = s^{-1} mod \ q$.
- Compute $u_1 = \mathcal{H}(m).w \ mod \ q$
- Compute $u_2 = r.w \ mod \ q$
- Compute $v = (g^{u_1} g^{u_2} \ mod \ p) mod \ q$

The signature is valid if and only if v equals r.

2.3.4 Correctness of the Algorithm

Since, $g = h^{\frac{p-1}{q}} \ mod \ p$, it follows that $g^q \equiv h^{p-1} \equiv 1 \ mod \ p$ by Fermat's Little theorem. Since $g > 0$ and q is prime, g must have order q. The signer computes

$$s = k^{-1}(\mathcal{H}(m) + xr) mod \ q.$$

Thus,

$$k \equiv \mathcal{H}(m)s^{-1} + xrs^{-1} \equiv \mathcal{H}(m)w + xrw \ (mod \ q)$$

Since g has order $q(mod \ p)$ we have,

$$g^k \equiv g^{\mathcal{H}w} \equiv g^{\mathcal{H}w} y^{rw} \equiv g^{u_1} y^{u_2} \ (mod \ p)$$

Finally, the correctness of the algorithm follows from,

$$r = (g^k mod \ p) mod \ q = (g^{u_1} y^{u_2} mod \ p) mod \ q = v$$

Chapter 3

Proposed BlockV Architecture

3.1 Preliminaries

Definition 1 *User U is defined as the $\sqcap$ bit address, randomly sampled without replacement from user address space $\mathcal{A}$.*

The user U receives its public key pk_U, secret key sk_U, wallet W_U at the time of opening the account. Each user can select *Rider* or *Driver* as the member class. Selection of both at a time is prohibited in any circumstances. However, the classes are interchangeable under certain restrictions. An user is represented by $\sqcap+1$ bit address of which the first $\sqcap$ bit is sampled from $\mathcal{A}$ and the last bit denotes the member class. 0 represents *Rider* and 1 represents *Driver*. The user U hence receives two addresses $addr_{Ur}$, $addr_{Ud}$ for each of the two member classes *Rider* and *Driver* respectively. For each member class address, there exist two parameters- namely, Rating R and Status stat. R_{Ux} denotes the last awarded reputation score obtained by the user U as member class x. This scores are updated as per successful or unsuccessful transactions carried out by the user after opening the account. Unsuccessful transactions are those in which the user has been proven malicious for his mischievous activity. $stat_{Ux}$ is the status of U as member class x at any point of time after the account opening. For both the reputation score and status, $x \in \{r, d\}$ where r and d symbolizes member class *Rider* and *Driver* respectively. Default value for reputation score is *zero* where as default value for status is *Unavailable* for member class *Driver* and *available* for member class *Rider*. The user is allowed to change its address only when it is not participating in a ride. Any assigned user address can be reused i,e. reassigned to a new user, if the prior user closes his account. When an user transacts in BlockV platform, its user address will be included in the blockchain ledger.

Definition 2 *Blockchain platform BlockV is defined as the set of α bit addressed live contract instances randomly sampled from the un-utilized address space $\mathcal{A}_{\mathcal{A}}$, each of which creates a bijective mapping with (rider, driver) pair.*

The contract instances are randomly sampled from the set of all α bit addresses $\mathcal{A}_A$. A contract address is said to be *live* if the contract is currently being executed i,e the ride mapped to that contract is not completed or objected to be unsatisfied. We define a taken address space $\mathcal{A}_T$ which contains only the live contract instances. An unique contract address is always ensured for next sampling by moving the sampled address from $\mathcal{A}_A$ to the taken address space $\mathcal{A}_T$. A contract instance when destroyed, corresponding address is returned to $\mathcal{A}_A$ to ensure availability free address in $\mathcal{A}_A$ for any new contract instance.

Proposition 1 *Cardinality of Contract address space $\mathcal{A}_T$ is equal to the cardinality of the user address space $\sqcap$, provided there exists no available contract instance in $\mathcal{A}_A$ and the user accounts are not anonymous.*

There are certain restrictions imposed on the users to ensure fairness in the communication protocols. First one is that all the users can choose only one address from the user address space at any time instance. If we consider that every user is indulged in some ride, each will be assigned a contract address corresponding to those rides. This proves that α has to be greater than or equal to $\sqcap$. Now, all the users can choose only one class, *Rider* or *Driver* at any time instance. The second restriction is that switching from one class to another requires absence of open contract in the existing class account. Hence, if the user address space is completely consumed and every registered user opts for the *Driver* class, there can be at most $2^{\sqcap}$ contract address used. This concludes that α has to be equal to $\sqcap$.

Here we need the restriction of non-anonymous user accounts as if an user carries multiple addresses, then the address space can not accommodate the same number of users as previous. Suppose, the users are permitted to have at most k many addresses at a time. Then we can accommodate only $2^{\sqcap - log_2(k)}$ no of users which is less than the no of users represented using $\sqcap$ bits.

Definition 3 *A Ride is defined as an event which pairs up an user from Rider class and an user from Driver class to execute a contract in BlockV platform.*

A contract address is assigned to an user in class *Driver* and an user in class *Rider* shares the same contract instance if they agreed upon a *Ride*. An user from member class *Rider* is denoted as $\mathcal{R}$. Similarly, an user from member class *Driver* is denoted as $\mathcal{D}$. A *Ride* is confirmed when $\mathcal{D}$ creates a contract by locking a security deposit $\mathcal{S}_{Ud}$ and $\mathcal{R}$ joins the same contract with security deposit $\mathcal{S}_{Ur}$ and fare after the path for the ride and fare paid is verified.

Definition 4 *A Location is represented by Cartesian coordinate as detected by Global Positioning System(GPS) for a point on the earth surface.*

Definition 5 *A Route is defined as the array of locations, where the consecutive two locations are linearly connected by a path, width of which is sufficient to drive a car as per regulatory standard.*

The first element present the Route array is called the start location $L_{startUr}$ selected from distribution Υ_1 and the last element in the Route array is called the end or drop location L_{endUr} selected from distribution Υ_2 by *Rider R*. Υ_1 is the distribution of the rider's daily travel start location and Υ_2 is the distributions of the rider's daily travel end locations.

Definition 6 *Fare f is defined as the monetary value representing the cost of the ride taken by the rider R computed from the Route based on a formula as per regulatory standard.*

Fare is proportional to the path covered by the route where path is the cumulative distance between two consecutive linearly connected locations. The term *linear* ensures a straight forward fare computation by calculating Euclidean Distances. However, the physical path may not be truly linear but consists of several bends, twist and turns along with the linear paths. Such a nonlinear path is approximated by joining piece wise linear path segments as depicted in Fig. 3.1. These segments are created with the aim of least mean square error of the reconstructed path from the true, smooth path curvature.

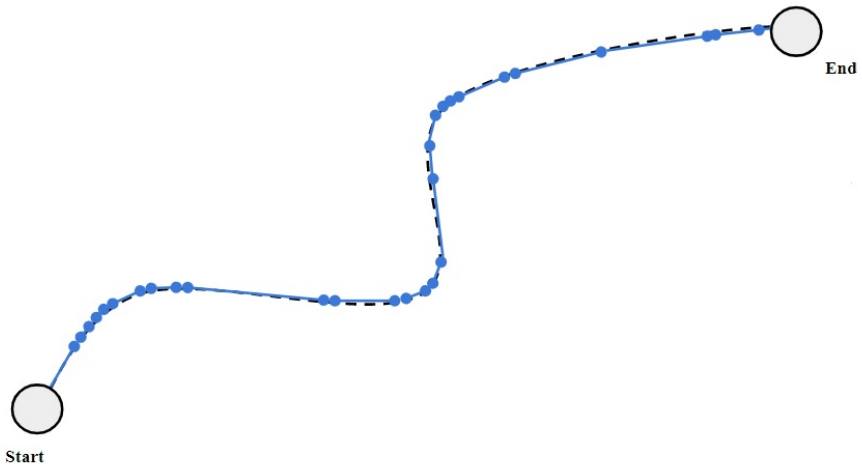


Figure 3.1: Piece-wise linear segmented continuous path

Given a pair of $(L_{startUr}, L_{endUr})$, there may exist more than one route satisfying the start and end locations. Since the Earth surface is spherical, we can generate infinite number of routes given two locations as stated. However, generating all the routes with the fares is highly inefficient considering the time complexity of the procedure and displaying all to the riders creates great inconvenience to them. These demands selecting a few routes given the locations. The most convenient routes are determined based on the trend of traffic conditions and updates of sudden occurrences of accidents, agitations, vehicle breakdown etc. We already have the technology with efficient GPS system to detect the traffic congestion.

This requires R to select any one route from the list of selected convenient routes. Each route in the list contains the array of locations and fare computed for that route. Selection of one route from a list is predominantly based on direct or indirect experience Υ_3 gained by the rider R that selects a specific route from the list of routes using feature values as weather, traffic, ongoing development works, man made disturbances, preference for known route etc as the application of human learning from environment. All the routes and the corresponding fares are kept in a decentralized Route Fare database (RFD) which is equally accessible by R and D.

Once the route is chosen, R requires send a join request to D along with the fare and a fixed amount of security deposit for *Rider* class. A join request received from R may not be accepted by D even if there is no mismatch of route and fare. The acceptance from driver D follows a distribution Υ_4 based on the daily schedule and preferred area of driving of D. The driver if rejects the join request has to prove that at the moment of receiving the join request he was located beyond a certain distance from the start location of the rider sending the request. This creates a circular geographical area centering the start location of the rider R. If any driver, after being present in this area, rejects the join request from R, he will be penalized. All other drivers whose locations are not inside the circular region, if approached by R with join request, may reject it without being penalized. The penalty in this case attracts monetary as well as reputation loss of the driver.

After a completed or semi-completed ride, R may not be satisfied with the service provided by D and trigger an objection. However a completed ride is not open to R or D for any operation. If a complain against D is lodged in BlockV platform, it should be undoubtedly resolved. A complain accompanies a $ProofofWrongRoute$ that makes sure D was present at a location not listed in the route contracted during the ride. Hence the presence or absence of D has to be easily verifiable. For the purpose of verification, we have considered the Road Side Units(RSUs) that can detect a car if it is nearby. A bijective mapping is assumed between a car and driver as at a particular time instance one implies the other.

Road Side Units in the context of Vehicular Adhoc NETwork (VANET) are the static points of V2X communications. RSUs are placed along the roads maintaining certain distances. They are meant for assisting the vehicular communication protocols. However, BlockV restricts itself to use the RSUs as the vehicle locating devices

Thus, if a car is detected by a RSU, from the log maintained in its database with time stamp , the mapped driver can be identified.

There are four major class of participants of the service *Ride*, namely $DRIVER$, $RIDER$, $BlockV$ and RSU. $DRIVER$s provide services and $RIDER$s accept the services by paying required amount as the cost of the service. $BlockV$ is the decentralized platform of blockchain which creates the fairness in the protocol running behind, starting from the offering of the services,issuing a contract to its closing. RSUs are the Road Side Units which acts as the validation helper. This platform also interacts with a secured decentralized route fare database RFD to compute and

display the possible route details and corresponding fare when queried.

3.2 High Level View

The process starts with the account opening. Each user(driver and rider) has to open an account to perform any task with BlockV. If the person opens the account as rider, he is already available or visible to the connected network. Else, he must have chosen to be the driver while opening the account. A driver needs to deposit a fixed security money to make himself available to the network every time from the Unavailable status it acquires in the subsequent procedures. Only the available riders and drivers are permitted to be involved in a new ride.

A rider if joins a driver for a ride, has to lock the fare and a fixed security deposit with it. The security deposit amounts can be the same or different. After the ride is confirmed, if the driver or the rider triggers the abort procedure, he needs to pay the entire amount he has locked for the ride. Else an available driver can withdraw his locked amount and abort which makes it Unavailable from available.

After the ride is confirmed, the driver and the rider creates a ledger of their locations with time stamp, in offchain. These ledger entries are only used while triggering a complain for unsatisfied ride with respect to the contracted path. We have defined an algorithm to ensure the fairness of the objection made by the rider if any. However,an innocent driver should not be penalized for a malicious rider and an innocent rider should not be harassed for a malicious driver.

Lastly, we define a procedure of completion for satisfactory service of ride. A complain triggered will conclude the ride at that position. A rider can not raise any complain for a ride which is already completed.This process continues till the members are having enough balance in their wallet. A member is allowed to change the member-type or de-activate the account only when no open ride is associated with the member's account. The flow of activities for an account holder is depicted in Fig. 3.2.

The architecture also includes the automated decentralized reputation score based system where, each driver or rider has to gain the score by completing the rides successfully without being penalized. The reputation system in the context of peer to peer network is explained in [2]. Another decentralized reputation system for marketplace is discussed in [43]. In BlockV, a successful completion of ride increases the reputation score where as an objection will decrease the score of the malicious party. The Abort procedure will also decrease the reputation score if the caller is in Busy state.

The users are allowed to change its address after it completes a ride. However, the reputation score it has gained till that point will not be changed. The user will only get a new address and corresponding public and secret keys for next ride keeping the member class selection and status unchanged.

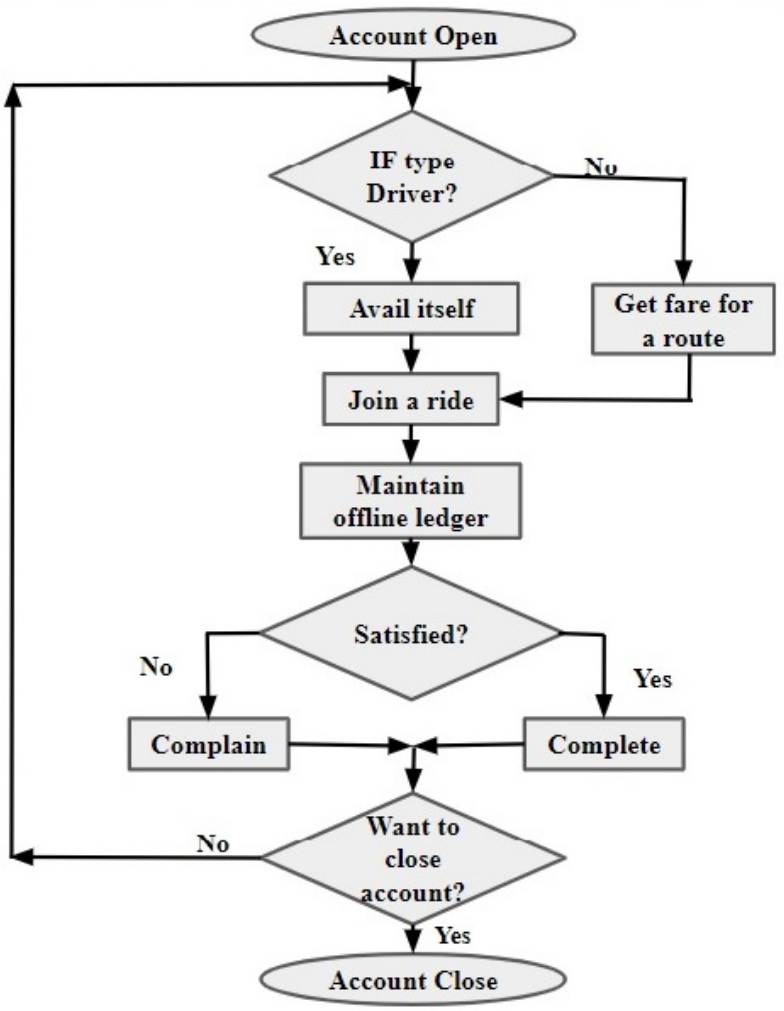


Figure 3.2: Flow Chart for an account holder

3.3 Procedures

The driver and the rider are the two key communicating entities. From the previous section we derive the sketch of communications as Fig. 3.3. Each of these communication protocol is described in this section sequentially.

3.3.1 Initialization-Account Open

Each user U has to open an account in BlockV platform to proceed further. U will be awarded with a $\alpha + 1$ bit address whose first α bit is chosen randomly from $\mathcal{A}$. Include the selected address $\mathcal{A}$ to $\mathcal{A}_U$. Generate (public key, secret key) with the chosen α bit address as seed input to KeyGen procedure.Initially users are registered

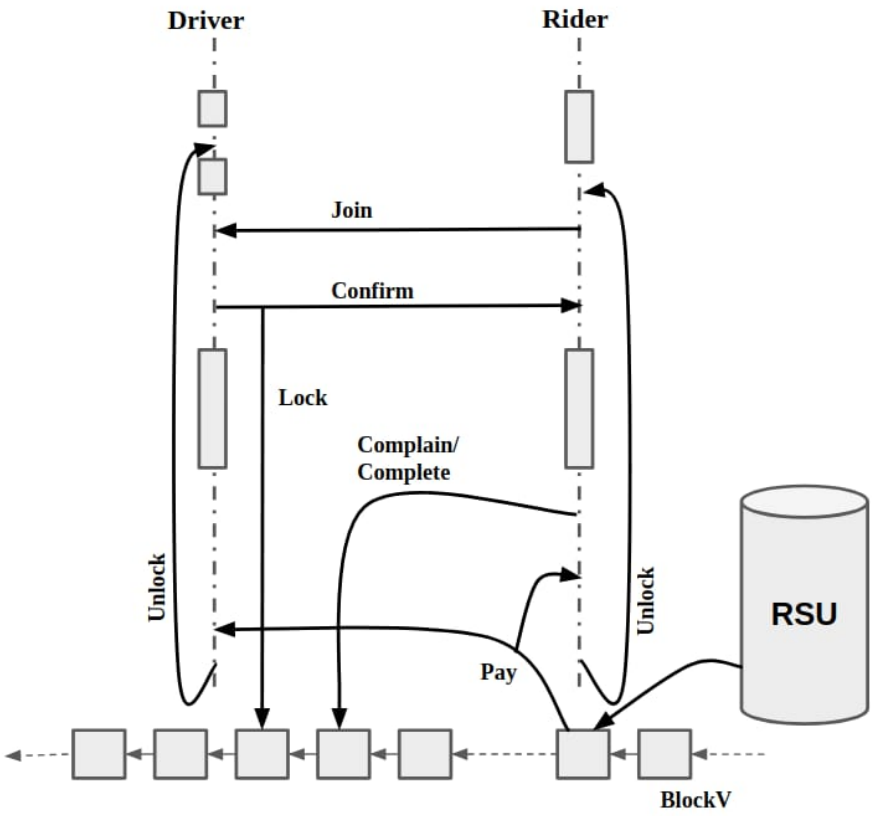


Figure 3.3: Communication overview

with one public key and one secret key. The public key generated by the KeyGen is en-queued in Arr_{PK} array and secret key generated is en-queued in array Arr_{SK}. The $addr_U$ is also put in Arr_{addr}.
Concatenate 0 and 1 with the chosen α bit address to generate $\alpha + 1$ bit addresses $addr_{Ur}$, $addr_{Ud}$ for the same user U to be represented as rider and driver respectively. Generate and initialize rating R_{Ur}, R_{Ud} for both the account types to zero. Status $stat_{Ur}$ for rider type is initialized to 'A' i,e. available and $stat_{Ud}$ for driver type is initialized to 'UA' i,e. unavailable. U can choose only one of the two account types and the corresponding status and reputation score will be live for updating in BlockV. An user account is associated with the wallet W_U and keeps track of its currently executing account by a contract address variable CC_U. Initially CC_U is NULL i,e, no contract address is attached with this user. BlockV platform allows each user to maintain a wallet for the transactions where money will be converted in crypto-currencies and kept in wallet.

Procedure 1: Account Open

Input : user U

Output: Public parameters : $addr_{Ur}$, $addr_{Ud}$, pk_U, R_{Ur}, $stat_{Ur}$, R_{Ud}, $stat_{Ud}$
Secret parameters: sk_U, W_U

1. $addr_U \leftarrow \$(\mathcal{A} - \mathcal{A}_U)$
2. s= size($\mathcal{A}_U$)
3. $\mathcal{A}_U[s+1] \leftarrow addr_U$
4. $ENQUEUE(Arr_{addr}, addr_U)$
5. $addr_count = 1$
6. $(pk_U, sk_U) \leftarrow KeyGen()$
7. $ENQUEUE(Arr_{PK}, pk_U)$
8. $ENQUEUE(Arr_{SK}, sk_U)$
9. $addr_{Ur} \leftarrow addr_U||0$
10. $addr_{Ud} \leftarrow addr_U||1$
11. $R_{Ur} \leftarrow 0$
12. $R_{Ud} \leftarrow 0$
13. $stat_{Ur} \leftarrow$A
14. $stat_{Ud} \leftarrow$UA
15. $W_U \leftarrow 0$
16. $CC_U \leftarrow NULL$

3.3.2 Key Generation

Consider a multiplicative group G of public keys of order p and generator g. Let sk_U be the n bit private key of user U. The public key pk_U is expressed as g^{sk_U}. The used secret keys are moved in to the set $\mathcal{SK}_{used}$

Procedure 2: KeyGen

Input :

Output: pk_U, sk_U

1. $sk_U \leftarrow \$(\{0,1\}^n - \mathcal{SK}_{used})$
2. $pk_U \leftarrow g^{sk_U}$
3. MOVE sk_U in $\mathcal{SK}_{used}$

3.3.3 Avail Driver

An account with type driver has a default initialization of status as Unavailable. The user U with status 'UA' is publicly not visible for the account type it currently holds. Hence, driver D should make itself available to utilize the facilities of BlockV. For

that purpose, D is required to lock a fixed amount of security deposit $\mathcal{S}_{Ud}$. Procedure LOCK with the input of $\mathcal{S}_{Ud}$ returns a contract address to D against which the money has been locked. If LOCK is successful, the status $addr_{Ud}.start_{Ud}$ is made 'A' i,e. Available. This enables the riders to find out this driver as ready for taking a ride.

Procedure 3: Avail Driver

Input : $addr_{Uy}$

Output: Public parameters : $stat_{dU}$ or $ABORT$

1. parse $addr_{Ud}= (addr_U, x)$
2. if $x = 1$ && $addr_U.start_{Ur}$ =UA && $addr_U.W_U >= \mathcal{S}_{Ud}$
3. $\quad addr_U.CC_U \leftarrow LOCK(addr_{Ud}, \mathcal{S}_{Ud})$
4. if $addr_U.CC_U \neq NULL$
5. $\quad addr_{Ud}.start_{Ud}$ =A
6. else $ABORT$

3.3.4 Route Select

This procedure can only be called by an user of account type rider. To start a ride, each rider requires to have a route and the corresponding fare. R has to select start location L_{start} from distribution Υ_1 and end location L_{end} from distribution Υ_2. Significance of distribution Υ_1 and Υ_2 is described in Chapter 2.

Procedure 4: Route Select

Input : $addr_{Uy}$

Output: (r, f) or $ABORT$

1. if $!VALIDR(addr_{Uy})$
2. $\quad ABORT$
3. $L_{start} \leftarrow \Upsilon_1\{\mathcal{L}\}$
4. $L_{end} \leftarrow \Upsilon_2\{\mathcal{L}\}$
5. $RF \leftarrow RFQD(L_{start}, L_{end}, addr_{Uy})$
6. $(r, f) \leftarrow \Upsilon_3\{(RF[i][0], RF[i][1]) : i \in [p]\}$
7. if $(r, f) \leftarrow NULL$
8. $\quad ABORT$

The procedure Route Select calls sub-procedure $RFQD$ with these two locations and get a list of convenient routes and their fares satisfying those locations. Rider R has to select any one route from that list, otherwise, the procedure aborts.

3.3.5 Join

Once R gets a route and its fare, creates a message by concatenating keyword 'CONF', current time stamp t_{now} and route r. This message along with the R's signature on it, fare f and security deposit $\mathcal{S}_{Ur}$, the address of the driver D, are sent to $CONNECTD$.

Procedure 5: Join

Input : $sk_{Ur}, pk_{Ur}, addr_{Ur}, addr_{Ud}, r, f, \mathcal{S}_{Ur}$
Output: $x \in \{CONFIRM, ABORT\}$

1. $msg = CONF||t_{now}||r$
2. $sig = SIGN(msg)$
3. $c \leftarrow 0$
4. parse $addr_{Ur} = (addr_U, x)$
5. if $addr_U.W_U > f + \mathcal{S}_{Ur}$ &&$x = 0$
6. $\quad c = CONNECTD(msg, sig, pk_{Ur}, f + \mathcal{S}_{Ur}, addr_{Ud}, addr_{Ur})$
7. if $c = 1$
8. $\quad addr_{Ur}.stat_{Ur}$ =B
9. $\quad x \leftarrow CONFIRM$
10. else $x \leftarrow ABORT$

If connected, the status of rider $addr_{Ur}.stat_{Ur}$ is changed to 'B' i,e. Busy. This procedure inherently changes the status $addr_{Ud}.stat_{Ud}$ of connected driver D to 'B'. The security deposit involved will be based on regulatory standard subjected to revision as per the economic condition.

3.3.6 During Ride

During the ride, starting from L_{start} to L_{end} both driver D and rider R may construct a ledger of locations with time stamp. The sequence of locations ideally reconstruct the route contracted between them. However, there may exist some deviations with in acceptance limit. Beyond that limit, if any move is complained by R, driver D is penalized. This ledger is used as the source of $ProoofOfWrongRoute$.

3.3.7 Complete

After the ride is complete and rider R is satisfied with the ride, R initiates the procedure *complete*. This unlocks the contract and release the money locked with it. $\mathcal{S}_{Ur}$ is returned to R and the rest is returned to D. Hence, the driver D will get its security deposit $\mathcal{S}_{Ud}$ back and the fare contracted as the remuneration.The successful completion awards both the rider and driver with the an increase in reputation scores. The status of the rider is made available and same for the driver is made unavailable

Procedure 6: *Complete*

Input : $addr_{Ur}, addr_{Ud}, C_{addr}$
Output: $c \leftarrow \{0, 1\}$

1. parse $addr_{Ud} = (addr_p, p)$
2. parse $addr_{Ur} = (addr_q, q)$
3. if $(addr_p.CC_p = addr_q.CC_q \;\&\&\; addr_p.CC_p = C_{addr})$
4. $\quad UNLOCK(C_{addr})$
5. $\quad addr_q.W_q \leftarrow addr_q.W_q + \mathcal{S}_{Ur}$
6. $\quad C_{addr}.lockedV \leftarrow C_{addr}.lockedV - \mathcal{S}_{Ur}$
7. $\quad addr_p.W_p \leftarrow addr_p.W_p + \mathcal{S}_{Ud}$
8. $\quad REMOVE\ addr_p.CC_p$ from $\mathcal{A}_T$
9. $\quad ADD\ addr_p.CC_p$ in $\mathcal{A}_A$
10. $\quad addr_p.CC_p \leftarrow NULL$
11. $\quad addr_q.CC_q \leftarrow NULL$
12. $\quad addr_{Ud}.statUd =$UA
13. $\quad addr_{Ur}.statUr =$A
14. $\quad addr_{Ud}.R_{Ud} \leftarrow addr_{Ud}.R_{Ud} + 1$
15. $\quad addr_{Ur}.R_{Ur} \leftarrow addr_{Ur}.R_{Ur} + 1$
16. $\quad c \leftarrow 1$
17. $c \leftarrow 0$

i,e both are set back to their initial status. Complete procedure unlocks a contract which completes the purpose of that contract instance. Hence, the instance is put to the available contract address space $\mathcal{A}_A$.

3.3.8 Complain

The rider R may raise a complain with $ProofOfWrongRoute$ if he is not satisfied with the ride. However, behavioral dissatisfaction is not considered here. The rider can only complain against the driver if during the ride the driver have taken a different route to reach the destination or it may not have reached the destination or the driver did not come to contracted pick up location. The proof shall include the evidence that shows the driver D was present at a location which is not included in the contracted path, even considering threshold distance from the nearest point in contracted route. A threshold distance is allowed in this case to appreciate the fact that the driver can not precisely maintain the contracted path, he has to adjust the position of the vehicle depending on the traffic. These adjustments are minute in nature hence can be separated from the cases where the driver has taken entirely new path to reach

the destination.

Procedure 7: *Complain*

Input : $powr, addr_{Ud}, addr_{Ur}, r, C_{addr}$
Output: $c \leftarrow \{0, 1\}$

1. $c \leftarrow 0$
2. parse $powr = (L, T)$
3. if $!EXISTD(L, T, addr_{Ud})$
4. $\quad penalty \leftarrow$R
5. else
6. $\quad$ parse $r = (l[1], l[2], ..., l[k])$
7. $\quad$ for $i \in [k] - 1$
8. $\quad\quad$ if $L_x > min(l[i]_x, l[i+1]_x)$ && $L_x < max(l[i]_x, l[i+1]_x)$ && $L_y > min(l[i]_y, l[i+1]_y)$ && $L_y < max(l[i]_y, l[i+1]_y)$
9. $\quad\quad\quad d \leftarrow DIST(L, l[i]) + DIST(L, l[i+1])$
10. $\quad\quad\quad$ if $d \leq DIST(l[i], l[i+1]) + \mathcal{DT}_2$
11. $\quad\quad\quad\quad penalty \leftarrow$R
12. $\quad\quad\quad\quad$ break
13. $\quad\quad\quad$ else $penalty \leftarrow$D
14. $UNLOCK(C_{addr})$
15. parse $addr_{Ud} = (addr_p, p)$
16. parse $addr_{Ur} = (addr_q, q)$
17. if $penalty \leftarrow$R
18. $\quad addr_p.W_p \leftarrow addr_p.W_p + C_{addr}.lockedV$
19. $\quad addr_{Ud}.R_{Ud} \leftarrow addr_{Ud}.R_{Ud} + 1$
20. $\quad addr_{Ur}.R_{Ur} \leftarrow addr_{Ur}.R_{Ur} - 1$
21. if $penalty \leftarrow$D
22. $\quad addr_q.W_q \leftarrow addr_q.W_q + C_{addr}.lockedV$
23. $\quad addr_{Ud}.R_{Ud} \leftarrow addr_{Ud}.R_{Ud} - 1$
24. $\quad addr_{Ur}.R_{Ur} \leftarrow addr_{Ur}.R_{Ur} + 1$
25. $REMOVE\ addr_p.CC_p$ from $\mathcal{A}_T$
26. $ADD\ addr_p.CC_p$ in $\mathcal{A}_A$
27. $addr_p.CC_p \leftarrow NULL$
28. $addr_q.CC_q \leftarrow NULL$
29. $addr_{Ud}.statUd =$UA
30. $addr_{Ur}.statUr =$A
31. $c \leftarrow 1$

The rider R should initiate procedure *Complain* with a location L and timestamp T. This (L, T) pair is sent to RSU checking whether D exists close to location L around time T. If no, it can be concluded that R have raised a false objection. Hence R is penalized. If yes, D is located at that position at that time. Hence, it is required to ensure that the location is out of the contracted route. To do so, we

need to compute the distance from L to each piece-wise linear component of the *route* provided x coordinate of L is in between the two x coordinates of the end points of the linear component. Similar condition applies for y coordinate. If the location found, check whether the sun of the distances of L from the end points of linear segment is less than the length of path segment plus $\mathcal{DT}_2$. This if satisfies proves the driver D is falsely objected as guilty, hence rider R will be penalized. For all other cases, driver D will be penalized. All penalties involve both monetary and reputation loss. Reputation score will decrease by one and the security deposit will be transferred to the other party.

3.3.9 Abort

Procedure 8: *Abort*

Input : $addr_{Ux}, addr_{Up}, sig, pk_{Ux}$

Output: $c \leftarrow \{0, 1\}$

1. if $VER(\text{"}ABORT\text{"}, sig, pk_{Ux})$
2. if $addr_{Up} \neq NULL$
3. parse $addr_{Ux} = (addr_x, x)$
4. parse $addr_{Up} = (addr_p, p)$
5. if ($addr_x.CC_x = addr_p.CC_p$ && $x \neq p$ &&$addr_{Ux}.stat_{Ux}$ =B &&$addr_{Ux}.stat_{Ux}$ =B)
6. $UNLOCK(addr_x.CC_x)$
7. $addr_p.W_p \leftarrow addr_p.W_p + addr_x.CC_x.lockedV$
8. $addr_{Ux}.RUx \leftarrow addr_{Ux}.RUx - 1$
9. $addr_{Up}.RUp \leftarrow addr_{Up}.RUp + 1$
10. if $x = 1$
11. $addr_{Ux}.statUx$ =UA
12. $addr_{Up}.statUp$ =A
13. else
14. $addr_{Ux}.statUx$ =A
15. $addr_{Up}.statUp$ =UA

An *Abort* called by R or D with status 'B' i,e. Busy attracts penalty. Penalty is same as described in Procedure 7. Otherwise, an Abort call by an Available D will make itself Unavailable and unlocks the security deposit to his wallet. Abort call by an Available rider R does not require any changes.

3.3.10 Get New Address, Public-Private Keys

The user can only change its address, public key and corresponding secret key, he has no active contract i,e. he is not engaged in a ride. Selection of The address and the

keys are similar to the initial selection procedures. If the queues reach the maximum memory limit $\mathcal{B}$, one element is removed by procedure $DEQUEUE$. New selected elements are put in those queues by procedure $ENQUEUE$.

Procedure 9: *ChangeIdentity*

Input : $addr_U$
Output: $c \leftarrow \{0, 1\}$

1. if $addr_U.CC_U = NULL$
2. $addr_{prev} = addr_U$
3. If $addr_count = \mathcal{B}$
4. $a = DEQUEUE(Arr_{addr})$
5. REMOVE p from $\mathcal{A}_U$
6. $p = DEQUEUE(Arr_{PK})$
7. $s = DEQUEUE(Arr_{SK})$
8. MOVE s in $\mathcal{SK}_{used}$
9. $pk_U, sk_U = KeyGen()$
10. $ENQUEUE(Arr_{PK}, pk_U)$
11. $ENQUEUE(Arr_{SK}, sk_U)$
12. $addr_U \leftarrow \$(\mathcal{A} - \mathcal{A}_U)$
13. s= size($\mathcal{A}_U$)
14. $\mathcal{A}_U[s+1] \leftarrow addr_U$
15. $addr_{Ur} \leftarrow addr_U||0$
16. $addr_{Ud} \leftarrow addr_U||1$
17. $ENQUEUE(Arr_{addr}, addr_U)$
18. $UPDATE_REP(addr_{prev}, addr_U)$
19. Select random time T_{rand}
20. PAUSE($T_{rand} + T_{hold}$)
21. Publish new address as identity.
22. $c \leftarrow 1$
23. else $c \leftarrow 0$

3.3.11 Deactivate

To deactivate one account the only requirement is absence of open contract associated with the account. This condition will ensure that no driver can close its account in between a ride and deny the service. An account holder when triggers this procedure, all the address allotted to this account, will be freed to reassign again for some new account opener. Similarly the private keys are moved back to $\mathcal{SK}_{used}$ for reuse.

Procedure 10: *Deactivate*

Input : $addr_{Ux}$

Output: $c \leftarrow \{0, 1\}$

1. parse $addr_{Ux} = (addr_U, x)$
2. if $addr_U.CC_U = NULL$
3. while $addr_count > 0$
4. $p = DEQUEUE(Arr_{addr})$
5. REMOVE p from $\mathcal{A}_U$
6. $s = DEQUEUE(Arr_{SK})$
7. MOVE s in $\mathcal{SK}_{used}$
8. $addr_count = addr_count - 1$
9. $c \leftarrow 1$
10. else $c \leftarrow 0$

3.3.12 En-queue and De- Queue

$ENQUEUE$ procedure enlists the input element at the end of the input queue. $DEQUEUE$ procedure removes the top element of the queue. Please note that, we have assigned the elements first and then en-queued at the end of the queue. Hence the top most element is the oldest.

3.4 Sub-Procedures

3.4.1 LOCK

Procedure 11: *LOCK*

Input : $addr_{Ud}, d$

Output: C_{addr}

1. parse $addr_{Ud} = (addr_U, x)$
2. if $d = \mathcal{S}_{Ud}$ && $x = 1$ && $addr_U.CC_U = NULL$
3. $c \leftarrow \$(\mathcal{A}_A - \mathcal{A}_T)$
4. $s = size(\mathcal{A}_T)$
5. $\mathcal{A}_T[s + 1] \leftarrow c$
6. $c.lockedV \leftarrow d$
7. $addr_U.W_U \leftarrow addr_U.W_U - d$
8. else $c \leftarrow NULL$

This procedure takes address of D $addr_{Ud}$ and the value to be locked as the input and returns the contract address with which the value is locked at the end of

the procedure. Parse $addr_{Ud}$ and check if the last bit is 1 to ensure that the user account type is driver. If confirmed, a contract address c is randomly sampled from $(\mathcal{A}_A - \mathcal{A}_T)$. c is enlisted in $\mathcal{A}_T$. This ensures uniqueness in selection. The value associated with this contract $c.lockedV$ is assigned with input d and the same is deducted from wallet of the user, $addr_U.W_U$. For all other cases, c is assigned to NULL.

3.4.2 JOINLOCK

Procedure 12: $JOINLOCK$

Input : $d, C_{addr}, addr_q$

Output: $c \in \{0, 1\}$

1. if $C_{addr} \in \mathcal{A}_T$ && $d = \mathcal{S}_{Ur}$
2. $\quad C_{addr}.lockedV \leftarrow C_{addr}.lockedV + d$
3. $\quad addr_q.W_q \leftarrow addr_q.W_q - d$
4. $\quad c \leftarrow 1$
5. $c \leftarrow 0$

This procedure takes the amount d, contract address C_{addr} where to lock the amount and the user address $addr_q$. If the C_{addr} is taken i,e. $\in \mathcal{A}_T$, then add d with the previously locked amount with that contract $C_{addr}.lockedV$ and deduct the same from wallet of that user. If all the steps are executed properly, outputs 1 else 0.

3.4.3 VALIDR

Procedure 13: $VALIDR$

Input : $addr_{Ux}$

Output: $TRUE$ or $FALSE$

1. parse $addr_{Ux} = (addr_U, x)$
2. if $x = 0$ && $addr_{Ux}.stat_{Ux}$ =A
3. $\quad$ output TRUE
4. output FALSE

Parse the address input this procedure $addr_{Ux}$, as $(addr_U, x)$ where $addr_U$ is α bit sequence. The last bit x if equals to 0 and the status $addr_{Ux}.statUx$ of the user U is 'A', then U is validated as rider. Note that there exist another valid status 'B' for rider. It is the context of the governing procedure, that requires this specific criteria.

3.4.4 RFQD

Given a start location L_{start} and end location L_{end}, this procedure returns a list of route and fare RF satisfying the locations. The procedure selects only the routes which starts at L_{start} and ends at L_{end} from a route database RD where all the routes are stored. For each entry in that set, CalcFARE returns the corresponding fare. The route and its computed fare is kept in the 2-D storage RF.

Procedure 14: RFQD

Input : $L_{start}, L_{end}, addr_{Uy}$
Output: RF

1. $R = \{\{l\}_{<k>} : l[0] = L_{start}, l[k-1] = L_{end}, k \in \mathcal{N}\}$
2. for $i, r \in R$
3. $\quad RF[i][1] = r$
4. $\quad RF[i][0] = CalcFARE(r)$
5. output RF

3.4.5 CONNECTD

This procedure establishes the link between rider R and driver D. The procedure takes six inputs. First three inputs are a message msg, signature sig of rider R on msg and public key pk_{Ur} of R. Procedure VER verifies the sig on msg using pk_{Ur}. If verified, parse msg as word w, time stamp t, route r. If w matches with word 'CONF' and the route is validated, extract fare f from input amount fd excluding the security deposit for rider $\mathcal{S}_{Ur}$. Parse $addr_{Ud}$ as $(addr_p, p)$ and $addr_{Ur}$ as $(addr_q, q)$. If the f matches with r, driver acceptance acc is taken from distribution Υ_4. If D accepts and JOINLOCK is successful with the call of fd, contract address $addr_p.CC_p$ linked to the account with base address $addr_p$ and base address of the rider R, $addr_q$, contract address of D is shared and assigned to R and statuses $addr_{Ur}.stat.Ur$ and $addr_{Ud}.stat.Ud$ of R and D respectively are assigned to 'B' i,e. Busy. Completion of these steps assigns c as 1 denoting success. For all other cases, check if D didn't accept and D attracts penalty as determined from procedure IFPENALTY. If so, rating $addr_{Ud}.R_{Ud}$ of the driver D is decremented by 1. The contract linked to D is unlocked. The money released with this shall be exactly the security deposit $\mathcal{S}_{Ud}$ of the driver D. This $\mathcal{S}_{Ud}$ is credited to the wallet $addr_q.W_q$ of the rider R. This concludes the penalty involved in this procedure. Now for the case of penalty, the contract address shall be included in the free contract address space by removing it from taken contract address space $\mathcal{A}_T$. Linked contract address $addr_p.CC_p$ of D is assigned NULL. Status $addr_{Ud}.statUd$ of D is made 'UA' i,e. unavailable. This second case assigns c as 0 denoting failure in establishing the link.

Procedure 15: $CONNECTD$

Input : $msg, sig, pk_{Ur}, fd, addr_{Ud}, addr_{Ur}$

Output: $c \in \{0, 1\}$

1. if $(VER(msg, sig, pk_{Ur}))$
2. parse $msg = (w, t, r)$
3. if $w = CONF$ && $VALIDROUTE(r)$
4. $f \leftarrow fd - \mathcal{S}_{Ur}$
5. parse $addr_{Ud} = (addr_p, p)$
6. parse $addr_{Ur} = (addr_q, q)$
7. if $((y \leftarrow FCHECK(r, f))$ && $(!q)$ && $(p))$
8. $acc \leftarrow \Upsilon_4\{TRUE, FALSE\}$
9. if $((acc)$ && $JOINLOCK(fd, addr_p.CC_p, addr_q))$
10. $addr_q.CC_q \leftarrow addr_p.CC_p$
11. $addr_{Ur}.stat.Ur =$B
12. $addr_{Ud}.stat.Ud =$B
13. $c \leftarrow 1$
14. else if $(!acc)$ && $(k \leftarrow IFPENALTY(r[0], addr_{Ud}, L_{nowUd}))$
15. $addr_{Ud}.R_{Ud} \leftarrow addr_{Ud}.R_{Ud} - 1$
16. $UNLOCK(addr_p.CC_p)$
17. $addr_q.W_q \leftarrow addr_q.W_q + \mathcal{S}_{Ud}$
18. $REMOVE\ addr_p.CC_p$ from $\mathcal{A}_T$
19. $addr_p.CC_p \leftarrow NULL$
20. $addr_{Ud}.statUd =$UA
21. $c \leftarrow 0$
22. else $c \leftarrow 0$

3.4.6 VALIDROUTE

The rider is supposed to join the driver with a path and the corresponding fare. A malicious rider may mutate some elements in the path and send the join request to the driver. The driver shall verify that the route is valid and physically exists. Hence, from the given route, parse the route as the start location L_{start}, an array of locations l' and the end location L_{end}. We get the set R from Route-Fare database filtering all the routes with L_{start} and L_{end}. The given route r is valid if $r \in R$.

Procedure 16: $VALIDROUTE$

Input : r

Output: $c \in \{TRUE, FALSE\}$

1. parse $r = L_{start}, l', L_e$
2. $R = \{\{l\}_{<k>} : l[0] = L_{start}, l[k-1] = L_{end}, k \in \mathcal{N}\}$
3. if $r \in R$
4. $\quad c \leftarrow TRUE$
5. else $c \leftarrow FLASE$

3.4.7 FCHECK

Procedure 17: $FCHECK$

Input : r, f

Output: $c \in \{TRUE, FALSE\}$

1. if $f = CalcFARE(r)$
2. $\quad c \leftarrow TRUE$
3. $c \leftarrow FALSE$

This procedure takes input a route r and a fare f and returns TRUE if f matches with r. FCHECK calls procedure CalcFARE with route r. If CalcFARE returns the same as f, FCHECK returns TRUE else, FALSE.

3.4.8 CalcFARE

Procedure 18: $CalcFARE$

Input : r

Output: f

1. $f \leftarrow F_{fixed}$
2. for $i, l \ \in RANGE(size(r) - 1)$
3. $\quad f \leftarrow f + p_1 \times DISTANCE(l[i],\ l[i+1])$
4. output f

CalcFARE calculates fare f given a route r. Consider fixed fare be F_{fixed}. f is initialized with F_{fixed}. For each consecutive location pair $l[i]$ and $l[i+1]$ in r compute the DISTANCE between them and add to f cumulatively. There exist $size(r) - 1$ many pairs in r. Hence the for loop runs for $size(r) - 1$ many times.

3.4.9 IFPENALTY

Procedure 19: $IFPENALTY$

Input : $L_{startUr}, addr_{Ud}, L_{nowUd}$

Output: $c \leftarrow \{TRUE, FLASE\}$

1. if $DISTANCE(L_{startUr}, L_{nowUd}) < \mathcal{DT}_1$
2. $\quad c \leftarrow TRUE$
3. $c \leftarrow FALSE$

IFPENALTY checks if the location L_{nowUd} of driver D currently is within distance threshold $\mathcal{DT}_1$ away from start location of route $L_{startUr}$. If yes, the procedure outputs TRUE else FALSE.

3.4.10 EXISTD

Procedure 20: $EXISTD$

Input : $L, T, addr_{Ud}$

Output: $c \leftarrow \{0, 1\}$

1. $RSU_{id} \leftarrow 0,\ MinD \leftarrow UINT_{MAX}$
2. $\forall RSU_i \in RSU_{DB}$
3. $\quad d \leftarrow DISTANCE(RSU_i, L)$
4. $\quad$ if $(d < MinD)$
5. $\quad\quad MinD = d$
6. $\quad\quad RSU_{id} = i$
7. $\forall entry_i \in RSU_{id}$ log with $t \ni (T \pm \Delta t)$
8. if $addr_{Ud} \in RSU_{id}log$
9. $\quad c \leftarrow 1$
10. $c \leftarrow 0$

Given a location L, timestamp T and address $addr_{Ud}$ of driver D EXISTD returns 1 if D is identified to be present in location L at T. We have assumed RSUs to be present every where forming a zone for each. Any car hence driver if steps in the zone will be recorded in the governing RSU with the time stamp. The first step is to find out the RSU nearest to L. For all RSU_i in RSU database RSU_{DB} check if the distance is minimum than previously encountered distance in the for loop. If yes, record it. At the end of the for loop, we receive the unique identification no id for the closest RSU. We consider a time slot with mean T and variance ΔT. Address log of RSU_id is checked for all the entries with time stamp in between $T + \Delta T$ and $T - \Delta T$. If any address found equal to $addr_{Ud}$, return 1. Else, return 0.

Chapter 4

Security Model

The architecture of BlockV described in Chapter 3 assumes the existence of Blockchain as the backbone of the model. For the purpose of security analysis of this architecture, we have assumed that blockchain is secured i,e. the consensus algorithms are working efficiently and the immutability properties hold. Given the blockchain is secured, the money locked in blockchain is secured hence the payment involved in the system is secured. The definition of the reputation score as a state variable in blockchain makes any changes in it noticeable by all the peers. Hence any unauthorized changes will be marked. Similarly we have defined the status of *Rider* and *Driver* as the state variable where the opposite party can check one's present status from blockchain. This makes the communications secured by choosing appropriate conditions for each procedure.

Proposition 2 *BlockV achieves the payment$fairness$ and ride$fairness$ assuming the blockchain, RSU and Route database RD are secure and the interaction with these entities are secure.*

Proof:

The sub-procedure $CalcFARE$ as discussed in Procedure 18 computes the linear path traveled and the corresponding fare by multiplying the cumulative path by a non negative fare component. The algorithm takes the route r which is a set of locations, forming the path the rider wants to travel. for each consecutive entries in r, $DISTANCE$ calculates the euclidean distance. We have added a fixed component of fare with this to cater the expenses of cost of establishment, toll taxes, maintenance cost etc. Given a path, computation of fare is available to all the peers in the network. Hence, the fare can be verified by all. So there exists a linearly proportional component as well as fixed component in the fare distribution. This ensures *payment $fairness$*.

The scheme offers the rider to come up with an objection before triggering the *Complete* procedure. We have considered both the cases where the driver is the malicious and tries to follow a route beyond contract or a malicious rider tries to

cheat the driver by accusing him with a false objection. *Complain* procedure as discussed in 7 handles these situations and penalizes the malicious party. Data (L, T) as the proof of wrong route, supplied by the rider with the objection, is considered for the further analysis. The driver must be located at the position at the time mentioned by the rider for any further checking. Assuming time delay associated with each electronic devices involved in the measuring units, we allow a grace period on the time mentioned. Hence, the driver shall be located at that position within $(T \pm \Delta t)$.

The data provided by the RSUs are taken into consideration to locate the driver. Since, we have assumed that the RSUs are secured, the data being used to locate the driver, is not tampered in any case. If the driver can not be located there, we conclude the rider to be malicious and end the procedure with some formalities including the charging of penalties, modification of reputation score. But if the driver is located, the next check shall be the correctness of the allegation.

From the practical point of view, a driver can not precisely maintain the contracted locations, but requires to adjust the positions to avoid clashes with the other vehicles. However, this adjustments are minute for example the database considers the bending radius of a curved path to be 10m but actually the driver may take 10.5m or 9.5m. Thus, mathematically if the actual locations are compared with the contracted locations, we will find deviations. So we have included a grace in distance also. While checking the location mentioned in the contracted path or not, we include the grace distance as distance threshold $\mathcal{DT}_2$ and check if the mentioned location lies in the range of the locations contracted. The list of locations create a piece wise linear route and we check This ensures the false penalizing the driver will not take place. This ensures $ridefairness$. □

The key aspects of the security notions in BlockV along with the anonymous property are described in subsequent sections.

4.1 Privacy involved in BlockV

Definition 7 *Privacy in BlockV is defined as the unlinkability of the transaction records in blockchain ledger.*

The users are identified by an address. If an user transacts four times a day, the rides can easily be linked by the user address and ride details like time, start and drop location can be fetched from the ledger. If the user is following the almost same riding time and the same drop locations, analyst can gain some information regarding the user's lifestyle. For example, if the analysis says that the drop location is "XYZ" institute for Monday to Friday around 10 a.m. and he takes another ride from that institute around 6 a.m.,one can assume that he is a stuff of that institute.

Hence the objective is that, looking in to the ledger, no one shall be able to link two rides taken by an user. However, the usages of the procedures for creating

unlinkable transactions are optional.

The transaction information is BlockV contains the addresses of the rider and the driver, a value and the ride details. The data in this transaction is not kept in the blockchain directly. Only the hash of the data is kept as the route details can be very large if the path has many bends and turns even if short distanced. The blockchain data can be downloaded by any peer. Hence each transaction details are open to all without any barrier. The blockchain platform proposed here shall be permissionless as a permissioned blockchain will need the existence of a central authority for access control. That will again raise a question on trust. Thus we cannot restrict a malicious user from getting the blockchain data or looking into the ledger.

The address written on the ledger is the primary concern of the user as the user can be identified uniquely by that address. Although the user details like name, contact details is not shared over the blockchain platform, the user can easily be personified and hence tracked. Thus we propose a solution where we can create a number of identities for an user. We define a bound $\mathcal{B}$, such that, any user can have at most $\mathcal{B}$ number of identities at a time.

We define the queues in Procedure 1 for each of the user address, public key and secret key. In this procedure, the account is registered with the inital keys and address. hence the $addr_count$ is set to 1. If any user wants to change its identity he can use the Procedure 9 and generate new public key, secret key and address. If the current address count is more than the bound, then one element in queue has to be removed to accommodate the new one. We use the Least Recent Used method of elimination to remove the top most elements from each queue. The current address $addr_U$, public key pk_U and the secret key sk_U are always set to the newly selected values. Hence we can generate and change the user identity.

The bound $\mathcal{B}$ is applied to the queue only to hold the address or keys against the user account. BlockV platform does not allow an user to set back to his earlier address or keys, still keeps the previous $\mathcal{B}-1$ many data attached. The reason is, if the previous address or keys are immediately rejected by this user, the elements will go back to their own allotment sets namely address will be removed from the used set $\mathcal{A}_U$ and the secret key will be removed from $\mathcal{SK}_{used}$. This makes them immediately alive for new allotments. For a network with maximum allowed user address N_u, the probability of selecting one address be $\frac{1}{N_u}$. An user can keep up to $\mathcal{B}$ many addresses. Hence the probability that the same address is being selected is $\frac{1}{N_u^{\mathcal{B}}}$.

For a small network this value may not be negligible hence we need to increase the value $\mathcal{B}$. Hence this is one kind trade-off where either the network has to be large enough to have all the equal valued selected addresses with negligible probability, or increase the bound $\mathcal{B}$ i,e. the holding capacity of the users. This ensures uniqueness is all subsequent address or keys selection for a single user. These indirectly creates a time gap of rejection of one address and next use of that.

This scheme ensures a new address or new keys every time when the user triggers the Procedure 9. This procedure is absolutely optional for the users to use as the

address and keys will always be linked to values obtained during registration phase. Hence the procedures can be freely accessed without using the procedure 9.

4.2 Linkable Reputation for Privacy Preserving Blockchain

Before discussing the linkable reputation scores, we need to first drive into the Ethereum Virtual Machine storage architecture[26]. Refer to Fig 4.1

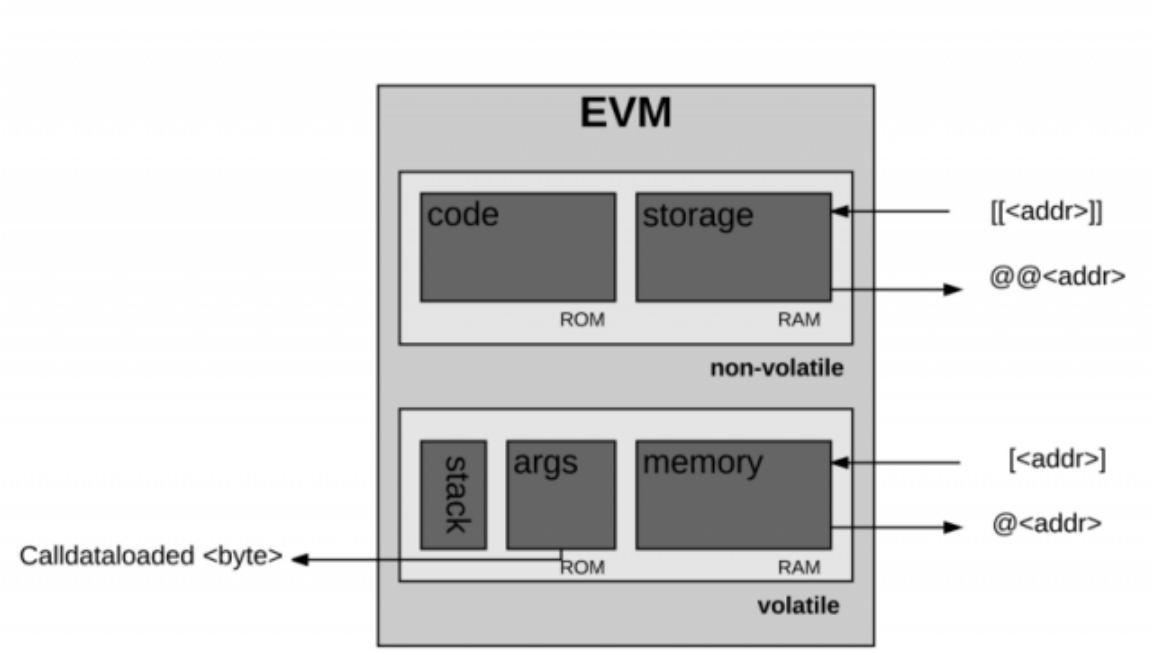


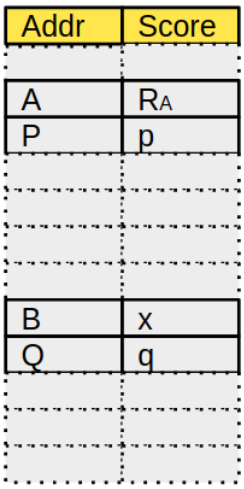
Figure 4.1: EVM Storage Overview

In the EVM we have three locations to store data - stack, memory, storage. Each of these storage locations plays important role in execution model of EVM. Stacks are used for computations where the operands for instructions are taken from stack and the result is also stored in stack. The Memory is the storage area which is created during function calls in smart contract. It stores temporary data such as function arguments local variables and return values. The memory is volatile. The next is Storage which is a persistent key-value store that maps keys to values.

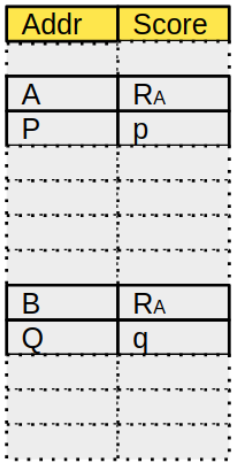
In solidity, we define mappings from user address to reputation score. For the time being, we are assuming that the users can not change their identities. Thus, Key-value storage will be created in the Storage location of EVM corresponding to

this mapping. Thus, for all available keys space for respective values will be assigned.

Now, when we introduce the anonymity in users' identification, we require to change the user address but we need to keep the mapping fixed i,e. we need to map the new address to the old reputation score. Once a mapping is defined, the map computes the maximum required space for its execution and initializes all the values to zero and keys are arranged in order. In our Procedure 9 we maintain a queue of last recent $\mathcal{B}$ no of addresses. Hence, we never delete the immediate previous address of an user when he changes its identity.



(a) Storage of Reputation Score before Identity Change



(b) Storage of Reputation Score after Identity Change

Figure 4.2: Change in Reputation Score with the change in Identity

Refer, Fig 4.2a where, user U was using address A and has gained reputation score R_A. Now the user has changed its address to address B. If B was not initialized before to any user, it will contain zero value as reputation score, otherwise it may contain any positive value, say x. From the address queue maintained in the user account, the user U can get the address A even if it has changed to address B. The address B is not published yet. At this step, the user can put the old reputation R_A corresponding to the value of address B as in Fig 4.2b.

The reputation score for the old address shall not be updated to zero for any unforeseen attack on EVM data storage as any new update may be monitored by other users but if the previous storage of reputation is made zero, the changes can be linked easily.

Once the procedure is completed the the user is ready to publish its new address B. The whole process is secured as the changing identity is an account level operation and not included in ledger. Until the new address B is published it is not possible

for other accounts to trace the change in address. Since we have also changed the reputation score from x to R_A, which can be monitored by malicious user, we have put the procedure in hold for time $T_{hold} + T_{rand}$.

We define $T_{hold} = \frac{1}{F_{IC}}$ where F_{IC} is the number of identity change per unit time. T_{rand} is a random time decided by the user for that operation. So, the procedure holds for time $T_{hold} + T_{rand}$, malicious user will not be able to link the update in reputation with address change as, by that time on an expectation half of the actual identity change will take place to hide the new address B. Hence the malicious user will not be able to detect the link between the two changes.

Chapter 5

Performance analysis of BlockV

We have implemented BlockV on Ethereum private and test networks(testnet). The contract address for KOVAN testnet is mentioned below for live interaction.

5.1 PC Configuration

We have used the PC with-

- Ubuntu 18.04 LTS, 64-bit operating system
- x64-based processor Intel(R) Core(TM) i5-8300H CPU @2.30GHz
- RAM - 8.00GB(7.85 GB usable)

5.2 Private Network

Gas Cost

Table 5.1 shows the gas required for each key procedures and the corresponding values in USD.We can see that the initial contract deployment attracts around 45 US cents. This is the highest value among all. For all other cases cost per transactions are within 2 US cents. All the USD values are based on the fact 1 ether= 141.7 USD as on 1st April, 2019.

Fig. 5.1 shows the Time/Transaction Vs. Number of Miner. The time per transaction decreases if the number of miners is increased. This is because of the fact that all the miners are sharing one physical machine where the experiment is set up and with the increase in the number of miners the probability that the transaction is successfully executed, increases. This describes the trend.

Fig. 5.2 shows the transaction time Vs. number of path segments. The time per transaction where computation over the path segments is involved, monotonically

Table 5.1: Table of gas Cost

Description	Gas/Transaction	USD/Transaction
Contract Deploy	1561271	0.44246
Account Open	57429	0.01854
Load Wallet	21808	0.00618
Abort when not busy	8045	0.00228
Abort when busy	14421	0.00409
Join	25065	0.0071
Complain	22460	0.00637
Complete	21894	0.0062

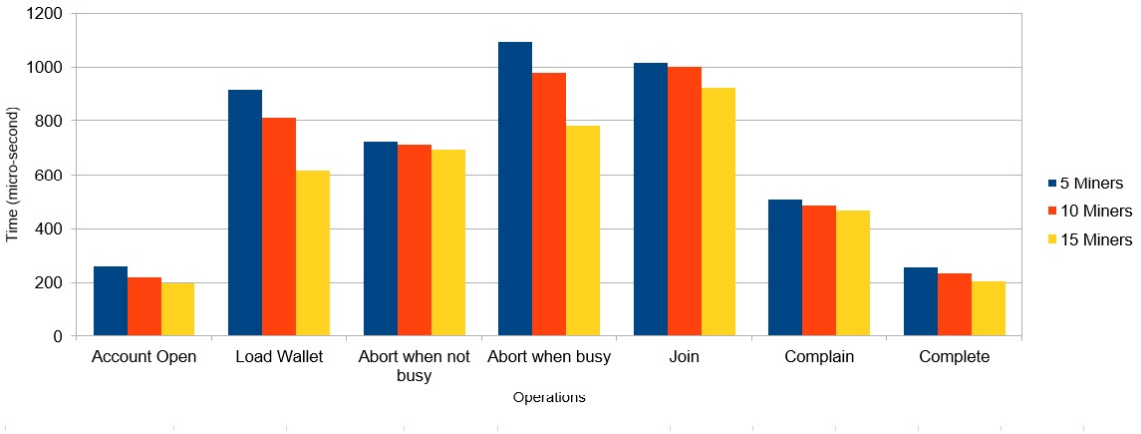


Figure 5.1: Time per transaction Vs. No of Miner

increases with the increase in number of path segments as the time involved in computation is directly proportional to the path.

5.3 Test Network

We have deployed the prototype in KOVAN testnet of Ethereum. The code is available at the contract address 0x78FCc0357745704c01FEC006039CF6E3b78b3678 in the KOVAN testnet of Ethereum.

It has to be noted that the data of the ride has to be kept in decentralized database and the digest will be kept in the BlockV. However, in the experiments, we have ignored the time taken for fetching data from the database. We have also made further simplification for the ease of implementation the distance check in *Complain*

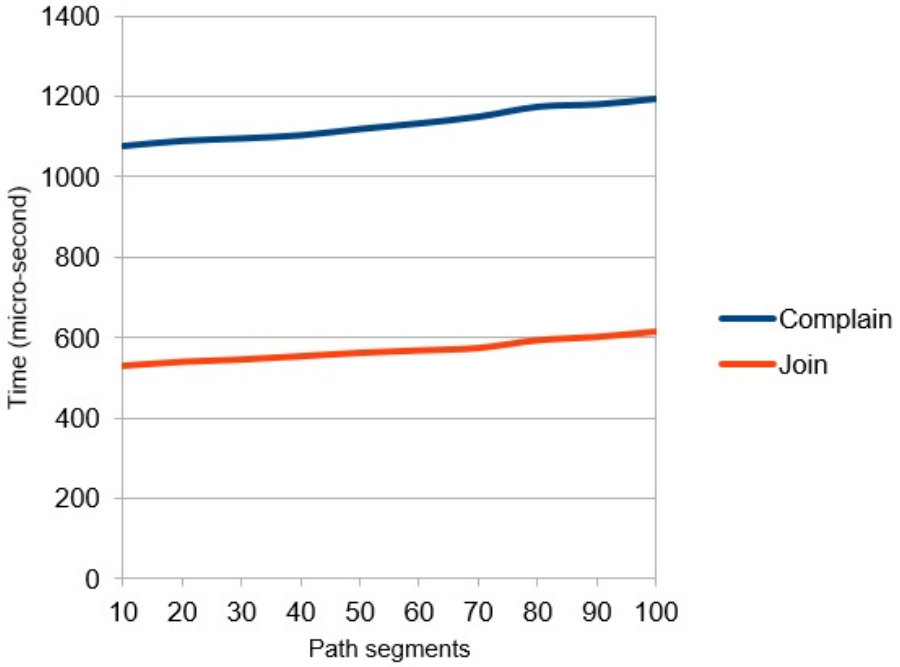


Figure 5.2: Time per transaction Vs. Path Segment

procedure. The Procedure 7 has been updated as per the simplification as the solidity language does not provide any support for complex mathematics and float values.

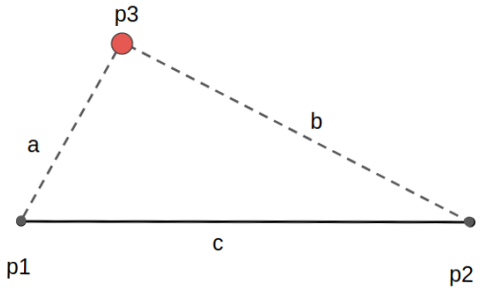


Figure 5.3: Path representation during complain

In the last stage of complain procedure it is required to detect whether the location provided with the objections is in the neighborhood of the path contracted. Let any two consecutive locations in the contracted path be $p1$ and $p2$ and the location mentioned with the complain procedure is $p3$. Let $a = DIST(p1, p3)$, $b = DIST(p2, p3)$ and $c = DIST(p1, p2)$ as in Fig 5.3.

We define the driver to be in the neighborhood of the path is equivalent to be in

the neighborhood of any path segments. Let the deviations allowed to the driver from the contracted path be p. Hence, a driver to be detected as present in the neighbour hood of the contracted path, $a + b \leq c + p \implies a^2 + b^2 + 2ab \leq c^2 + p^2 + 2cp$.

Consider the product of two numbers a and b, $ab \leq max(a^2, b^2)$ and $min(a^2, b^2) \leq ab$. Also, for all practical purposes, the deviation p allowed will be less than the value c. Hence the above inequality reduces to $a^2 + b^2 + 2 \times min(a^2, b^2) \leq 3 \times c^2 + p^2$. This approximation has been used while implementing the smart contract in solidity language.

Chapter 6

Conclusion and Future Work

The BlockV architecture described in this thesis meets the two fairness goals along with the privacy goal. The reputation score associated with every user makes the scheme more user oriented where the reputation score represents the overall successful ride the user has performed. The scheme also supports the anonymous behavior of the users where an user can transact with multiple addresses and still maintains the reputation score associated with it.

With out the anonymity, the user accounts are uniquely mapped to the reputation system. The mapping is strict to the address representing the account. This restricts the users from being anonymous.

An anonymous user shall be allowed to change its identity before the connected network without changing or resetting its gained reputation score and wallet values. The requirement is fulfilled with this architecture.

A heavy system with a large number of peers will be prone to reach consensus slower than the required pace. In that case, that validator set can be selected as a subset of all the peers within a fixed region based on the contracted path. This will make the process faster as in this case we will be considering only a small subset of the peers.

Similarly, when the rider wishes to join a ride, he may get to see available only those drivers who are within a fixed radius of the rider's start location. The rider may be offered several fares for his opted path by the drivers and the rider may select any one of them based on the fare offered and the reputation score gained by the driver. Hence, the views of the riders can be restricted to only nearest drivers with good rating based on the rider's own rating.

The problem of dispute resolution is hard in all practical cases as the fixing a route is only a coarse adjustment of a path. The fine tuning has to be done during the driving. So, mathematically it is hard to define the difference between the fine tuning and detouring. Hence it is convenient to define a neighborhood of the contracted path and allow this fine tuning. The locus of neighborhood as described in this paper forms an ellipse contracted or chopped from both side along the path segment end

points. We, can also consider a flat neighborhood around the contracted path such that membership values of all locations in the selected range are higher for contracted path than the detoured path.

The security deposit amounts shall be dynamically adjustable as per the money values which requires the deposit amount to be a function of the relevant dependencies. The fare shall include the route the driver has to travel to reach the rider's start location.

In this thesis we have shown that BlockV platform maintains the transparency in the ride and the overall system is stable with respect to the increasing number of participants and their frequency of rides. This is also profitable as the transaction costs are few cents. Also, the in-build reputation system is secured in blockchain. The platform allows the user U to behave as a rider as well as driver based on his personal requirements without any switch in account. This makes the system user friendly. In future we would like to make the system anonymous where the user may change its appearance but still mapped to its old reputation. We have showed that this scheme in ethereum platform is highly scalable and the cost of each transaction is very less.

Bibliography

[1] Biswas, K., Muthukkumarasamy, V.: Securing smart cities using blockchain technology. In: 2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS). pp. 1392–1393. IEEE (2016)

[2] Buchegger, S., Le Boudec, J.Y.: A robust reputation system for mobile ad-hoc networks. Tech. rep. (2003)

[3] Chiu, J., Koeppl, T.V.: Blockchain-based settlement for asset trading. The Review of Financial Studies 32(5), 1716–1753 (2019)

[4] Chohan, U.W.: A history of dogecoin. Discussion Series: Notes on the 21st Century (2017)

[5] Crawford, M.: The insurance implications of blockchain. Risk Management 64(2), 24 (2017)

[6] Dannen, C.: Introducing Ethereum and Solidity. Springer (2017)

[7] Dorri, A., Kanhere, S.S., Jurdak, R.: Towards an optimized blockchain for iot. In: Proceedings of the second international conference on Internet-of-Things design and implementation. pp. 173–178. ACM (2017)

[8] Dorri, A., Steger, M., Kanhere, S.S., Jurdak, R.: Blockchain: A distributed solution to automotive security and privacy. IEEE Communications Magazine 55(12), 119–125 (2017)

[9] Dziembowski, S., Eckey, L., Faust, S.: Fairswap: How to fairly exchange digital goods. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 967–984. ACM (2018)

[10] Eberhardt, J., Heiss, J.: Off-chaining models and approaches to off-chain computations. In: Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers. pp. 7–12. ACM (2018)

[11] Eiza, M.H., Shi, Q., Marnerides, A.K., Owens, T., Ni, Q.: Efficient, secure, and privacy-preserving pmipv6 protocol for v2g networks. IEEE Transactions on Vehicular Technology 68(1), 19–33 (2019)

[12] ETSI, T.: 102 637-2, intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of co-operative awareness basic service. ETSI, Sophia Antipolis Cedex, France (2010)

[13] Fauri, D., de Wijs, B., den Hartog, J., Costante, E., Zambon, E., Etalle, S.: Encryption in ics networks: A blessing or a curse? In: 2017 IEEE International Conference on Smart Grid Communications (SmartGridComm). pp. 289–294. IEEE (2017)

[14] Ferrag, M.A., Maglaras, L., Argyriou, A., Kosmanos, D., Janicke, H.: Security for 4g and 5g cellular networks: A survey of existing authentication and privacy-preserving schemes. Journal of Network and Computer Applications 101, 55–82 (2018)

[15] Haferkorn, M., Diaz, J.M.Q.: Seasonality and interconnectivity within cryptocurrencies-an analysis on the basis of bitcoin, litecoin and namecoin. In: International Workshop on Enterprise Applications and Services in the Finance Industry. pp. 106–120. Springer (2014)

[16] Hirai, Y.: Defining the ethereum virtual machine for interactive theorem provers. In: International Conference on Financial Cryptography and Data Security. pp. 520–535. Springer (2017)

[17] Jacobovitz, O.: Blockchain for identity management. The Lynne and William Frankel Center for Computer Science Department of Computer Science. Ben-Gurion University, Beer Sheva (2016)

[18] Kalodner, H.A., Carlsten, M., Ellenbogen, P., Bonneau, J., Narayanan, A.: An empirical study of namecoin and lessons for decentralized namespace design. In: WEIS. Citeseer (2015)

[19] King, S., Nadal, S.: Peercoin–secure & sustainable cryptocoin. Aug-2012 [Online]. Available: https://peercoin. net/whitepaper () (2018)

[20] Korpela, K., Hallikas, J., Dahlberg, T.: Digital supply chain transformation toward blockchain integration. In: proceedings of the 50th Hawaii international conference on system sciences (2017)

[21] Leiding, B., Vorobev, W.V.: Enabling the vehicle economy using a blockchain-based value transaction layer protocol for vehicular ad-hoc networks. URL: https://uploads-ssl. webflow.

com/5a4ea18a81f55a00010bdf45/5b69e53263e2a6076124ecbe_Chorus-Mobility-WP–v1. 0.1. pdf (2018)

[22] Li, L., Liu, J., Cheng, L., Qiu, S., Wang, W., Zhang, X., Zhang, Z.: Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. IEEE Transactions on Intelligent Transportation Systems 19(7), 2204–2220 (2018)

[23] Lin, I.C., Liao, T.C.: A survey of blockchain security issues and challenges. IJ Network Security 19(5), 653–659 (2017)

[24] Liu, H., Zhang, Y., Yang, T.: Blockchain-enabled security in electric vehicles cloud and edge computing. IEEE Network 32(3), 78–83 (2018)

[25] Liu, Z., Ma, J., Jiang, Z., Zhu, H., Miao, Y.: Lsot: a lightweight self-organized trust model in vanets. Mobile Information Systems 2016 (2016)

[26] Mayoya Tudonu: A deep dive into the ethereum virtual machine (evm) (2019), `https://www.mayowatudonu.com/blockchain/deep-dive-into-evm-memory-and-storage`, [Online; accessed 27-June-2019]

[27] Mettler, M.: Blockchain technology in healthcare: The revolution starts here. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom). pp. 1–3. IEEE (2016)

[28] Mohseni-Ejiyeh, A., Ashouri-Talouki, M.: Sevr+: Secure and privacy-aware cloud-assisted video reporting service for 5g vehicular networks. In: 2017 Iranian Conference on Electrical Engineering (ICEE). pp. 2159–2164. IEEE (2017)

[29] Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system (2008)

[30] O'Dair, M., Beaven, Z., Neilson, D., Osborne, R., Pacifico, P.: Music on the blockchain (2016)

[31] Oham, C., Jurdak, R., Kanhere, S.S., Dorri, A., Jha, S.: B-fica: Blockchain based framework for auto-insurance claim and adjudication. In: Proceedings of the IEEE 2018 International Conference on BlockChain

[32] Ortega, V., Bouchmal, F., Monserrat, J.F.: Trusted 5g vehicular networks: Blockchains and content-centric networking. IEEE Vehicular Technology Magazine 13(2), 121–127 (2018)

[33] Panchalika Pal, S.R.: Blockv: A blockchain enabled peer-peerride sharing service. In: The 2nd IEEE International Conference on Blockchain. IEEE (2019)

[34] Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)

[35] Report, T.: Filecoin: A cryptocurrency operated file storage network

[36] Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation systems. Communications of the ACM 43(12), 45–48 (2000)

[37] Rivera, R., Robledo, J.G., Larios, V.M., Avalos, J.M.: How digital identity on blockchain can contribute in a smart city environment. In: 2017 International Smart Cities Conference (ISC2). pp. 1–4. IEEE (2017)

[38] Rowan, S., Clear, M., Gerla, M., Huggard, M., Goldrick, C.M.: Securing vehicle to vehicle communications using blockchain through visible light and acoustic side-channels. arXiv preprint arXiv:1704.02553 (2017)

[39] Shao, J., Lin, X., Lu, R., Zuo, C.: A threshold anonymous authentication protocol for vanets. IEEE Transactions on vehicular technology 65(3), 1711–1720 (2016)

[40] Sharma, P.K., Moon, S.Y., Park, J.H.: Block-vn: A distributed blockchain based vehicular network architecture in smart city. JIPS 13(1), 184–195 (2017)

[41] Sharma, P.K., Park, J.H.: Blockchain based hybrid network architecture for the smart city. Future Generation Computer Systems 86, 650–655 (2018)

[42] Singh, M., Kim, S.: Intelligent vehicle-trust point: Reward based intelligent vehicle communication using blockchain. arXiv preprint arXiv:1707.07442 (2017)

[43] Soska, K., Kwon, A., Christin, N., Devadas, S.: Beaver: A decentralized anonymous marketplace with secure reputation. IACR Cryptology ePrint Archive 2016, 464 (2016)

[44] Sun, J., Yan, J., Zhang, K.Z.: Blockchain-based sharing services: What blockchain technology can contribute to smart cities. Financial Innovation 2(1), 26 (2016)

[45] Tian, F.: An agri-food supply chain traceability system for china based on rfid & blockchain technology. In: 2016 13th international conference on service systems and service management (ICSSSM). pp. 1–6. IEEE (2016)

[46] Vogelsteller, F., Buterin, V., et al.: Ethereum whitepaper. Ethereum Foundation (2014)

[47] Whitepaper: The future of human capital management - how individuals benefit from earning digital credentials

[48] Whitepaper: Media chaina de-centralised blockchain focused on the film industry

[49] Wikipedia contributors: Digital signature algorithm — Wikipedia, the free encyclopedia (2019), https://en.wikipedia.org/w/index.php?title=Digital_Signature_Algorithm&oldid=901940368, [Online; accessed 27-June-2019]

[50] Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper 151, 1–32 (2014)

[51] Xia, Q., Sifah, E., Smahi, A., Amofa, S., Zhang, X.: Bbds: Blockchain-based data sharing for electronic medical records in cloud environments. Information 8(2), 44 (2017)

[52] Zyskind, G., Nathan, O., et al.: Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops. pp. 180–184. IEEE (2015)