2023

Trustworthy Decentralized Last Mile Delivery Framework Using Blockchain

Ala' Alqaisi
University of Windsor

Trustworthy Decentralized Last Mile Delivery Framework Using Blockchain

By

Ala' Alqaisi

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2023

Trustworthy Decentralized Last Mile Delivery Framework Using Blockchain

by

Ala' Alqaisi

APPROVED BY:

F. Baki
Odette School of Business

X. Yuan
School of Computer Science

S. Saad, Advisor
School of Computer Science

January 23, 2023

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

The fierce competition and rapidly growing eCommerce market are painful headaches for logistics companies. In 2021, Canada Post's parcel volume peaked at 361 million units with a minimum charge of $10 per each. The Last-Mile Delivery (LMD) is the final leg of the supply chain that ends with the package at the customer's doorstep. LMD involves moving small shipments to geographically dispersed locations with high expectations on service levels and precise time windows. Therefore, it is the most complex and costly logistics process, accounting for more than 50% of the overall supply chain cost. Innovations like Crowdshipping, such as Uber and Amazon Flex, help overcome this inefficiency and provide an outstanding delivery experience by enabling freelancers willing to deliver packages if they are around. However, apart from the centralized nature of the Crowdshipping platforms, retailers pay a fee for outsourcing the delivery process, which is rising. Besides, they lack transparency, and most of them, if not all, are platform monopolies in the making.

New technologies such as blockchain recently introduced an opportunity to improve logistics and LMD operations. Several papers in the literature suggested employing blockchain and other cryptographic techniques for parcel delivery. Hence, this thesis presents a blockchain-based free-intermediaries crowd-logistics model and investigates the challenges that could harbor adopting this solution, such as user trust, data safety, security of transactions, and tracking service quality. Our framework combines a security assessment that examines the possible vulnerabilities of the proposed design and suggestions for mitigation and protection. Besides, it encourages couriers to act honestly by using a decentralized reputation model for couriers' ratings based on their past behavior. A security analysis of our proposed system has been provided, and the complete code of the smart contract has been publicly made available on GitHub.

DEDICATION

I would like to extend my sincere gratitude to my mentor Dr. Sherif Saad Ahmed, for his time and direction throughout this work. His encouragement, patience, and understanding have been priceless to me.

Secondly, I would like to thank my thesis committee members, Prof. Yuan and Prof. Baki, for putting time and effort into evaluating my work. Without their valuable advice and insightful suggestions, I would not have been able to complete this research successfully.

I am also immensely grateful to my children, Serine and Salman, for the love, kindness, and prayers that they have made so that I may achieve my goals.

Last but not least, I want to express my overwhelming appreciation to my parents, my family, and my friend Safiia for supporting me spiritually and mentally throughout writing this research paper and in my life in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

LMD	Last-Mile Delivery
BC	Blockchain
SC	Smart Contract
DAO	Distributed Autonomous Organization
PoW	Proof of Work
PoS	Proof of Stake
PBFT	Practical Byzantine Fault Tolerance
CFT	Crash Fault Tolerance
RTT	Monitoring round-trip time
ACL	Access Control List
DDoS	Distributed Denial of Service
PoD	Proof of Delivery
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
SDK	Software Development Kit
TPS	Transaction Per Second

CHAPTER 1

Introduction

1.1 Motivation

Last-mile delivery (LMD) refers to the last segment of the supply chain process that transfers goods from final distribution centers to customers' doorstep. Due to consumers' dispersed destination locations, last-mile delivery is the most costly and challenging stage. It contributes the largest share, approximately more than 53% of overall shipping cost [20]. In the last decades, the tremendous evolution of online shopping and the recent pandemic have been causing spectacular growth in the parcel shipping market across the globe. For instance, Canadian e-Commerce sales, which made up over $43 billion in 2018, are projected to increase by another 25% by 2023 to reach $55.4 billion [47]. As a result, Canada spent over $19.9 billion on last-mile delivery services in 2021 [63]. The market for these services is expected to increase in North America from 2021-2026, reaching $74.36 billion at a CAGR of 16.48% [81]. As e-Commerce sales have grown, customer preferences and expectations have been boosted; consumers need same-day shipping, free home delivery, real-time package tracking, and free returns. These new conditions pressure shipping professionals to consider alternative LMD solutions.

Meanwhile, the world has witnessed a rapid evolution of the sharing economy, which refers to a peer-to-peer activity of obtaining, giving, or sharing access to goods and services through e-applications. Logistics providers exploited the same concept to develop a new business model called Crowdshipping, which delegates parcel delivery tasks to a crowd of local, non-professional couriers for monetary compensation

using their vehicles or other transportation modes. The most common example of crowdshipping LMD platforms is within the food and restaurant industry, such as UberEats, Amazon Flex, SkipTheDishes, DoorDash and Instacart.

The crowdshipping framework differs from shipping via traditional employees dedicated to delivery service; crowd drivers may vary significantly in their time and detour flexibility. Some drivers may only want to make a slight detour from their original route, while others may be free and able to make multiple deliveries despite location and time. Therefore, crowdshipping maximizes logistics efficiency by downsizing the operational costs of package delivery, enhancing customers' flexibility to schedule deliveries with online shipment tracking, and reducing traffic and emissions [29].

However, while these applications make life easier for the customer, they make it harder for others. To begin with, the organizations that run these apps act as intermediaries and deduct unjustifiable high commission rates for managing the delivery process between the retailer and the buyer. For instance, SkipTheDishes and UberEats commission rates range between 20% and 30% of each order's value [24]. Secondly, giant companies' fierce competition and investments created a trend toward a monopoly and induced an uneven distribution of the welfare produced in the crowd-sourced delivery field, which forced small businesses to consider whether they could afford to continue playing in the delivery sector. Furthermore, most of these platforms are deployed on a centralized architecture, which could expose the system to data corruption, privacy breach, and single point of failures risks, resulting in monetary and credit damage. For instance, DoorDash was a victim of hackers who stole the information of 4.9 million of their customers, delivery workers, and merchants [14], while similar incidents happened with UberEats in 2020 [58].

From this perspective, there is a call to transform the present platforms or construct an alternative business model that overcomes these drawbacks. The possibility to respond to this call is the Blockchain, which may potentially made a vital contribution to the supply chain and logistics field due to its critical features, such as immutability, integrity, and confidentiality. Blockchain can create trusted decentralised applications and reduce their reliance on third parties by utilizing smart

contracts that define an automatic agreement between seller, courier, and consignee when transacting. Plus, distributed ledger guarantees transparency and avoids the risk of data tampering and infrastructure failure. Hence, this thesis aims to design a blockchain-based crowdshipping platform managed directly by retailers, customers, and couriers transparently and without intermediaries.

1.2 Problem Statement

Designing a decentralized LMD solution that is secure and trusted is urgently needed. To build a secure LMD system, we must look at the usage of blockchain and smart contracts. The scheme should provide a reliable approach to promote fairness and trust among the LMD stakeholders. The characteristic of such design based on current LMD and crowdshipping challenges are:

1. Support P2P delivery operations.
2. All logistic stakeholders can manage shipping and handle disputes.
3. Support Same Day Delivery, On-demand Delivery and Customized Delivery.
4. Ensure each participating entity in delivering a product have equal opportunity to participate without reserving collateral.
5. Establish trust using reliable reputation system that encourage good behaviours, block malicious actors, and support fairness.

1.3 Research Questions

The current study aims to design a decentralized crowdshipping model that connects the primary stakeholders without intermediaries. Answers to the following research questions are required to reach this goal:

- How can we utilize blockchain technology to provide a Crowdshipping system free of mediators without compromising the security and trust of the proposed platform?
- What are the limitations of similar work in the literature?
- What are the security and privacy challenges of presented schemas in literature and our proposed schema?
- How can we remediate the proposed schema's potential security threats?
- What techniques can leverage trust between system entities?

1.4 Thesis Contribution

The main contributions and the challenges the thesis aims to address can be summarized as follows:

- Propose a blockchain-based system to realize decentralized crowdshipping services that eliminates the need for a central authority or third-party involvement.
- Examine the potential attacks associated with the Blockchain design, the peer-to-peer architecture, and the crowdsourced delivery application.
- Investigate the defense tactics that could be used to strengthen the security of the proposed platform.
- Propose a reputation model for couriers based on their prior behaviors to inject trust between participants and discourage malicious behavior in the system. Additionally, contrasts it with the Average reputation model used by several platforms experimentally.
- Implement a proof of concept using Hyperledger Fabric, a real-world permissioned blockchain platform and conduct intensive experiments and performance evaluations in a test network.

1.5 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 explains the Blockchain principles necessary to comprehend the rest of this thesis and see the potential of Blockchain technology.
- Chapter 3 summarizes related work and presents an analysis and limitations of these works.
- Chapter 4 describes the approach and methods applied in this work.
- Chapter 5 presents an illustration of the design and implementation of our subject application, including the Data Model, Smart Contracts, API, and Query.
- Chapter 6 provides feature-based comparision with existing work in the literature, the reputation model test scenarios, and performance measurement's experimental setup and its results.
- Chapter 7 concludes the thesis and proposes directions for future work.

CHAPTER 2

Blockchain Fundamentals

This chapter illustrates the Blockchain fundamentals which will help understand the remainder of this thesis. First, explain the different types of Blockchain technology. Next, describe public key cryptography, hash function, digital signature, and consensus. Finally, analyze the Hyperleger Fabric blockchain architecture and the transaction flow, as it is the development environment of the subject in this study.

In 2009, Bitcoin was the first practical Distributed Autonomous Organization (DAO). DAO is an organization that aims to operate without centralized authority or control. Satoshi Nakamoto published a white paper titled "Bitcoin: Peer-to-Peer Electronic Cash System," which covered the concept and operation of the cryptocurrency system as a DAO [59]. Bitcoin addressed problems of decentralization by adding the following two innovative features to the system. First is data distribution to all nodes so the system network transparently monitors it. Since all system activity is watched and reviewed by all network participants, the system can operate in a secure state without any centralized governance. Second is the incentive structure for data set (block) processing. Blockchain is designed to provide financial compensation to the node that has successfully validated a new data set and added it to the existing data chain. Monetary compensation is funded from the profits generated through the operation of the encrypted currency system, which collects a certain percentage of the transactions between users. With such an ingenious structure, the cryptocurrency system can operate as a DAO.

2.1 Generic elements of a blockchain

Understanding the key components of a blockchain ecosystem and their various roles is crucial for diving deep into blockchain network transactions. The following list includes the essential elements of the blockchain:

- **Transactions**

  The smallest unit in the blockchain is a transaction. According to Lantz and Cawrey [46], a transaction refers to the transfer of value from one node to another. The blockchain makes it possible to send and receive information between network nodes. The exchange utilizes files that store data being transferred from one node to another and are made available to the entire network for verification. Then, a group of transactions is put together in blocks. Every node keeps a copy of the current blockchain and a log with a history of previous transactions.

- **Blocks**

  The blockchain data structure consists of a chain of blocks; each block is a compiled piece of data. Similar to a linked list, each block contains a reference to the preceding block, building a connection back to the chain's first block (also referred to as the genesis block). This reference is the block's header data hashed together. The hash represents a unique, fixed-size identifier that represents each block; no two blocks will have the same identifier. The elements listed in figure 2.1.1 collectively make up a block. The Block Header and Transactions are a block's two most crucial components since they provide the hash value. The block's overall size is known as the block size. The block's metadata is all contained in the block header, and the transaction counter tracks the number of transactions. Finally, all the Transactions are stored in the block.

- **Nodes**

  A blockchain node is a free, open-source, multi-platform runtime that enables developers to build different services. With the P2P protocol, nodes can ex-

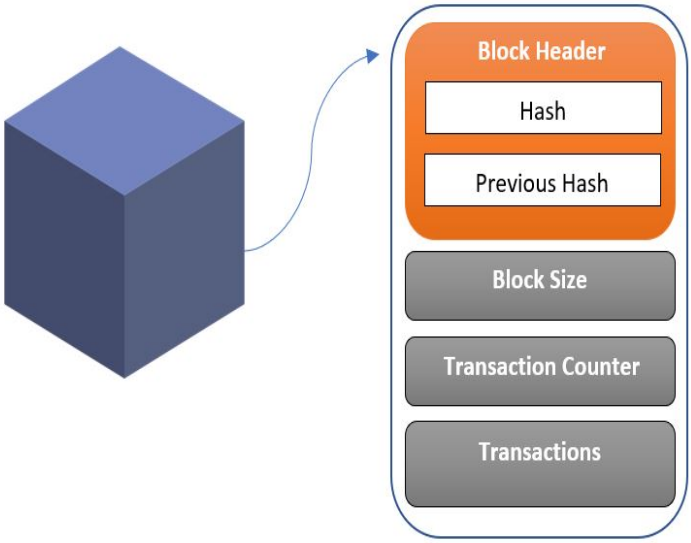

Fig. 2.1.1: Structure of a block

change data about transactions and new blocks with one another inside the network. Each node store a copy of the distributed ledger. Each network node is responsible for the accuracy and dependability of the data stored in the distributed ledger.

- **Distributed Ledger**

  A distributed ledger is a kind of database replicated, synchronized, and shared among the participants in a decentralized network. The distributed ledger keeps track of all interactions between network participants, such as exchanging assets or information [13].

2.2 Blockchain Types

There are four types of blockchain, public, private, consortium, and hybrid [75]. A public blockchain, referred to as permissionless, is open for everyone; anyone can join the network, create a transaction, validate transactions, or view previous transactions without requiring authorization. For instance, Bitcoin and Ethereum are

public blockchains that rely on a Proof-of-Work (PoW) consensus, which needs massive computational power to resolve the cryptographic challenges. On the contrary, private or permissioned blockchains mandate peers to obtain permission from federated authorities to enter the consensus protocol. Private Blockchain transactions are not publicly viewable in the network. Permissioned blockchain simplicity aids in reducing the necessary time for operations, and it offers more flexibility and increases efficiency compared to public blockchain [16]. Hyperledger, Corda, and BigchainDB are examples of permissioned blockchain platforms.

The hybrid Blockchain is a composition of the previous types; members determine who can participate in the blockchain or which transactions are made public. Hybrid Blockchain offers the speed of private blockchains combined with all the essential characteristics of a public blockchain, such as security, transparency, immutability, and decentralization. XDC is a project created and managed by XinFin that takes advantage of public and private blockchain [31]. The last type is consortium blockchain, known as a federated blockchain. It is pretty similar to private blockchains, but multiple organizations govern the platform instead of only a single organization. So, it aims to support cooperation to meet the constant challenges of a particular industry and overhaul transparency, accountability, and workflow to save money and time on development. TradeLens is an IBM/Maersk blockchain-based supply chain initiative that includes more than 175 organizations [78]. The project aims to create a single ecosystem that can bring together cargo owners, shipping companies, transportation providers, carriers, port facilities and terminals, customs, and other governmental organizations.

2.3 Public Key Cryptography

Cryptography is One of Blockchain's core that protects the user's identity and ensures any transaction's authenticity and integrity [17]. Cryptography applies mathematical algorithms to transmit data to control who can access and process this data [18]. While encryption is the process of encoding transmitted data to an unreadable format

that third parties cannot intercept. Public key cryptography utilized to validate transactions in the Blockchain network is based on an asymmetric key encryption scheme that uses public and private keys. The public key is shared over the network and known by all the network participants, while the private key must not become known to any other except the key holder. Any data encrypted with the public key can only be decrypted using the paired private key. Any data signed with the private key can be verified using the corresponding public key. Figure 2.3.1 illustrates a secure and encrypted network communication between two users, Alice and Bob. Alice encrypts the desired message using Bob's public key, yielding a ciphertext. Only Bob, who knows the corresponding private key, can decrypt the ciphertexts to obtain the original message.

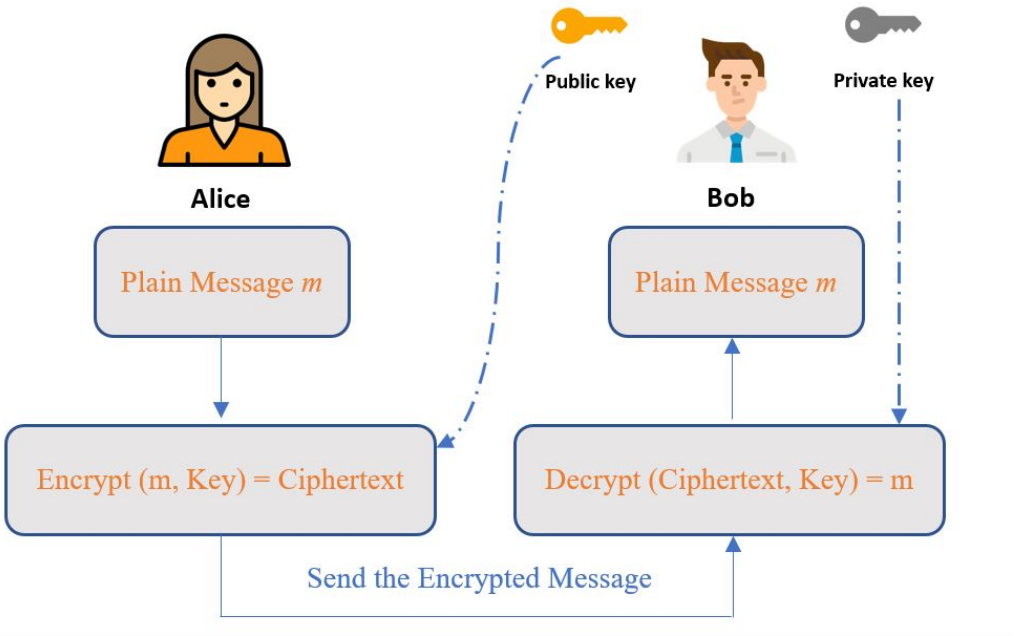

Fig. 2.3.1: Public Key Cryptography

2.4 Hash function

A hashing function is a one-way cryptographic algorithm that transforms and generates a fixed-size string out of input data of any length. The hash function is deterministic, meaning that a hash function's outcome is always identical. However, any

slight change in the input data changes the result completely. Hyperledger Fabric and Bitcoin Blockchains use the SHA-256 (Secure Hash Algorithm) hashing algorithm as one of the most robust hash functions. For example, the hash function, H = SHA-256, is implemented to hash the University of Windsor student number, **'110040153'** , which generates its associated hash value as follow:
'7335aa090755e3d06bb3a1276de85fa21136e8b559fa058062962e3e869c5bed'

2.5 Digital Signature

Digital Signature is a cryptographic proof system that can help establish trust. In Blockchain, the user uses the private key to sign the transaction, and the corresponding public key will help to authorize the transaction's sender. Data integrity, authentication, and non-repudiation are the three goals of digital signatures. In the preceding illustration 2.3.1, Bob can confirm that Alice's message was not altered in transit. An entirely new signature would result from any changes to the message. Bob may use Alice's public key to verify that Alice and no one else produced the digital signatures as long as Alice's private key is kept a secret. Unless her private key is hacked in some way after the signature has been created, Alice won't be able to later deny signing the document.

2.6 Consensus

Consensus algorithms in blockchains are used to decide the validity, the order of the next batch of transactions and to reach a mutual agreement on the present data state of the ledger . Depending on its type and applications, various consensus algorithms are used for blockchain. The most commonly used consensus algorithms are Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT) [36]. Proof of Work (PoW) is used in public blockchain (e.g., Bitcoin) to confirm a block creation. Before creating a block, a hash value for the unconfirmed transactions from the pool with a nonce is generated. A nonce is a random number used at most

once within a session. The network's miners attempt to guess the nonce in order to be able to get a hash value. Once a miner obtains the hash value, the transactions are confirmed, and all network participants can easily validate the block. The miner who guessed the nonce value and created the block receives bitcoins rewards.

On the other hand, in Proof of Stake (PoS), the miner is randomly selected based on the stake or minted cryptocurrency it possesses. The protocol locks the selected miner's stake until successfully adding the block. Any illegitimate attempt while adding the block, a penalty may be imposed and deducted from the already locked stake. Moreover, the selected miner charges some transaction fees for adding the block instead of getting a reward; hence, miners have no competition. The foremost advantage of the proof of stake algorithm is reducing the need for computational ability and hence a lower entry barrier for block generation rewards.

Private blockchain avoids the mining (computational) overhead in the public blockchain algorithms, where miners need to use computational power, time, and cryptocurrency. Instead, it handles the consensus among the users through state machine replication. PBFT (Practical Byzantine Fault Tolerance) is a Byzantine Fault Tolerance algorithm with low complexity and high practicality in distributed systems. The primary replica has to order transactions and obtain approvals from other replicas. When the network's replicas share messages to commit a block to the chain, a malicious replica may propagate a tampered block. However, the block which is considered valid by a maximum number of nodes is considered the valid one by the entire network. PBFT needs to receive approvals from honest nodes that exceed two-thirds of the total number of replicas to guarantee the security of the network [91]. Once satisfactory approvals are received, the primary replica commits the block and broadcasts it to the network.

2.7 Hyperledger Fabric Architecture

One of the most well-known distributed ledger frameworks is HyperLedger Fabric, created by the HyperLedger blockchain open-source project and supported by the

Linux Foundation. HyperLedger Fabric aims to develop apps or solutions with a modular architecture and plug-and-play components for membership and consensus. HyperLedger Fabric is a permissioned blockchain, and Table 2.7.1 lists its main components. There were several reasons for selecting Hyperledger fabric. Substantially, it is a non-cryptocurrency-based blockchain, meaning there is no POW algorithm and crypto mining in Fabric. Thus, it delivers high scalability and fast transactions. Hyperledger Fabric is an open source with detailed documentation and several implementation examples. Furthermore, it supports the minimum technical requirements to build the proof of concept.

Component	Description
MSP	Membership Services Provider (MSP) is implemented as a Certificate Authority to manage certificates used to manage and authenticate member identity and roles of all participants on the network. No unknown identities can transact in the Hyperledger Fabric network.
Peer	Peers are a vital element in the network that host ledgers and chaincode (smart contracts). An application interface, ledger data access, endorsement of transactions, and chaincode execution are all performed by a peer. Some peers can be endorsing peers which validate transaction requests from the client, commit the block received from the Orderer and update its ledger.
Orderer	Orderer are nodes which produces a block containing the endorsed transactions after sorting them according to the time they were received from peers, then distributes the blocks to all other peers in the network.
Client	Clients act on behalf of the system end-user by submitting transaction-invocation requests to the endorsers and broadcasting transaction proposals to the orderers.
Chaincode	Chaincode refers to the smart contracts used by Hyperledger Fabric. Chaincode is a program that holds the system's business logic and executed when predefined conditions are met [15]. When an application has to communicate with the ledger, the application invokes the Chaincode. Chaincode is deployed to all peers at the initialization stage of the fabric network.
Channel	Channels are a logical structure formed by multiple organizations to create a separate ledger of transactions. When configuring any channel, a set of policies must be agreed upon to govern the interactions between organizations and define the permission to invoke the chaincode deployed on this channel.
Organization	Organization is an entity that consists of a group of peers who have an identity (digital certificate) assigned by a Membership Service Provider.

Table 2.7.1: Major components of HyperLedger Fabric

After joining a channel in Hyperledger Fabric, peers maintain a copy of the ledger. The ledger is divided into two sections as illustrated in figure 2.7.1 . A blockchain data structure containing the blocks makes up the first part (of transactions). The second component, a world-state database, stores the most recent state following a block's commit. Upon successful validation, the peer commits the new block received from the ordering service into the ledger. The block is added to the blockchain, and

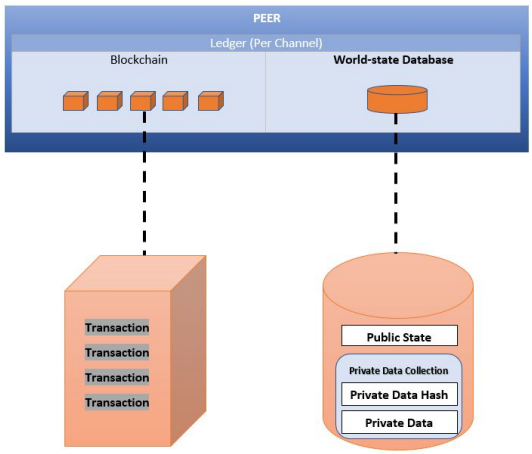

Fig. 2.7.1: HyperLedger Fabric Data Structure

each transaction is updated in the world-state. Most of the ledger is identical among peers inside a channel due to the consensus. However, Private Data is an exception, as only specific organizations store it in the world-state. In some cases, just a subset of the organizations requires to keep data private from other organizations on a shared channel. To meet this demand, Hyperledger Fabric introduces Private Data through the definition of data collection. All peers inside the subgroup can see the private data, while peers outside the subgroup will preserve a record of the private data hash as proof of data existence or for audit purposes. Each organization has an implicit data collection for private data by default.

Endorsement, Ordering, and Validation are the three stages of consensus in Hyperledger Fabric. Endorser nodes must endorse a transaction based on policy, such as (m out of n) signatures. The ordering phase accepts the approved transactions and consent to the order to be added to the ledger. Finally, the validation phase examines a block of arranged transactions to ensure accurate outcomes, including reviewing endorsement policy and double-spending. The current consensus algorithms in Hyperledger fabric are CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant) to support different trust assumptions of a particular deployment or solution [6].

The Hyperledger Fabric transaction flow process consists of eight phases, as outlined below and depicted in figure 2.7.2.

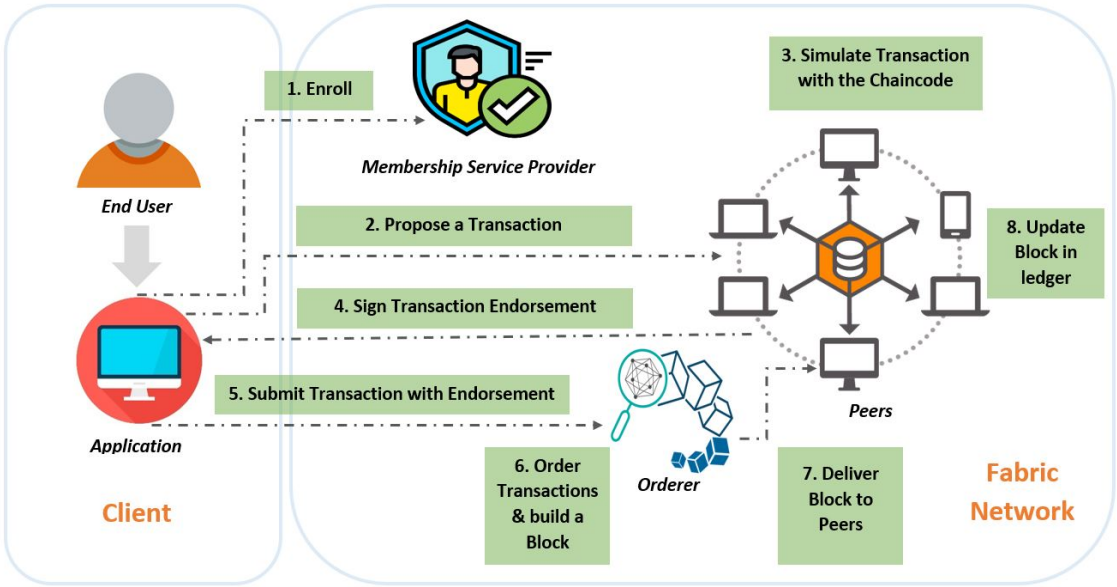

Fig. 2.7.2: Transaction Flow in Hyperledger Fabric

Step 1: The user enrolls in the MSP through an application, and the MSP issues them a User ID and a certificate.

Step 2: The user proposes a transaction to network peers.

Step 3: Endorsing peers who received the transaction from the user perform a validation check on the client's identity to ensure they are authorized for their request. The transaction is then simulated using the pre-deployed Chaincode.

Step 4: After successfully simulating the Chaincode, each peer gives the user their endorsement.

Step 5: The user gathers peer endorsements and sends them to the orderer.

Step 6: The orderer organizes the endorsed transactions received in the previous step in chronological order and constructs a block containing them.

Step 7: Orderer distributes the block to all the network's peers after sorting the endorsed transactions received from peers.

Step 8: Each peer updates its ledger by appending the new block to the prior block after receiving and verifying it. At this stage, the ledger is identical for all peers.

Hyberledger Fabric is one of the best-performing platforms currently available in transaction processing and transaction confirmation latency since it combines distinctive design features. First, it supports pluggable consensus methods to match

specific use cases and trust models. Second, it uses consensus mechanisms that don't need a native cryptocurrency to reward costly mining or power for executing smart contracts. The lack of a cryptocurrency minimizes certain key risks and attack vectors. The platform can be set up for nearly the same operational cost as any other distributed system because there are no cryptographic mining operations.

CHAPTER 3

Related Works

This chapter reviews related work found about LMD, and the application of Crowdshipping and blockchain in LMD. The main emphasis is on literature that supports leveraging blockchain to deliver physical assets and utilizing reputation-based systems.

3.1 Crowdshipping

Crowdshipping is a service inspired by crowd-sourcing. Crowdshipping selects freelancer drivers, who are the closest to the shipping route, to deliver the packages from the seller location to the last distribution center, or the customer location, using their vehicles for a particular compensation. Crowdshipping is known by different terms, such as Crowd-sourced delivery, LMD crowd-sourcing, Crowd logistics, and cargo hitching. Since crowd logistics platforms enable long-distance routing, tracking, and tracing of items and provide feedback on delivery and payment of services, they are referred to as the backbone of the logistics process [49]. In an effective implementation, crowdshipping benefits the community by decreasing the freight delivery trucks needed in urban and suburban regions. Additionally, it helps businesses save money on delivery while keeping the same level of service. This innovative design also offers social collaboration, allowing non-professional people to participate in LMD procedures and participate in a group activity that could ultimately result in a more sustainable community. If these advantages materialize, crowdshipping could result in a more economically and environmentally efficient system that eventually

improves everyone's quality of life [64].

Several studies consider the potential of crowdsourcing the delivery process of small parcels. Archetti et al. [7] were the first to model the issue of crowdsourced drivers in logistics networks by simulating the vehicle routing problem with occasional drivers. They introduced the potential of outsourcing some of the demand fulfillment to occasional drivers or in-store customers whose origin and destination almost match those of the packages and who are prepared to make a delivery on their way home. The article provides some preliminary study on the advantages of using crowdsourcing in logistics systems and encourages future investigation in this area.

Arslan et al. [9] consider a new dynamic variation of the pickup-and-delivery problem with time windows, where delivery tasks are matched to a specific group of ad-hoc drivers or a third-party backup fleet. An ad-hoc drivers may make several stops throughout their route to pick up and deliver several packages. Each package is picked up at its starting point and shipped to its final location. From an economic standpoint, Qi et al. [67] investigate and contrast the issue with conventional shipping. This work is the first attempt to conceptualize and evaluate a future crowdshipping system using analytical models and empirical parameter estimations. The study look at a network with many trans-shipment nodes or last-mile delivery terminals. Outbound deliveries are handled by shared-mobility drivers, while inbound shipments are transported by the logistics service provider's trucks. The authors contend that one of the key characteristics of shared mobility is its one-way, one-shot nature; a car begins an outbound trip by approaching its first demand location, and the service ends once it drops off the last package.

Pourrahmani et al.[64] provide an overview of the operational characteristics of crowdshipping platforms with empirical case studies to evaluate the service effects on each actor (e.g., seller, buyer, courier) in particular and society in general in terms of quality of life, cost, traffic externalities, and revenue. At the same time, Alnaggar et al.[3] analyze the current industry status of crowdshipping and provide a classification of available platforms based on their target markets, matching framework, and compensation strategies.

3.2 Blockchain Based LMD Systems

AlTawy et al. in [5] proposed a blockchain-based platform called Lelantos that focuses on enabling an anonymous customer to get delivery service of physical goods. Their scheme aim to hide the identity of the customers and their private information from any of the contractual parties (shippers) and prevent linkability between the customer and the merchant, which might be used for adversarial purposes. Using a web service, customers upload their real addresses to the blockchain in an encrypted manner, select at least two couriers, and redirect shipments between different couriers using a smart contract function without a trusted third party. Once the courier arrives at the customer's address, a secure hashing is utilized to prove the delivery. However, this scheme increases computing costs and complexity for buyers, who must select at least two delivery companies to ship their packages. Additionally, it enforces Point-of-No-Return as the cancellation is not addressed in their schema, and their PoD is not linked to the item of interest (package).

Hasan and Salah in [37] designed a blockchain-based solution of physical asset delivery and analyzed a Proof of Delivery system to trade and track sold items between two parties using automated payments through ethers. Each involved party deposits collateral equal double the package price that each party risks losing if any behaves maliciously. Their protocol ensures asset handover verification through a key exchange provided by the seller to the courier and the buyer. The proposed POD solution ensures accountability, timeliness, dependability, and auditability. The blockchain-based solution offers tamper proof logs for auditability and traceability. To ensure integrity, the scheme uses the InterPlanetary File System (IPFS) hash of the signed terms and conditions form in the smart contract. Plus, their work demonstrates accountability by using keys and hashes for verification of the correct customer. Refunds and cancellations are also handled to protect the interests of the seller, the customer, and the transporter. The suggested solution also makes use of a Smart Contract Attestation Authority (SCAA) to ensure that the code adheres to the terms and conditions approved by the involved parties. However, the scheme

does not fit a crowdshipping environment since it require double package price deposit per order. Besides, the relationship between the key and the package is not stated. Consequently, the courier can easily tamper with the asset to be delivered.

Ngamsuriyaroj et al. in [60] developed a blockchain-based package delivery system that would help improve security by ensuring data accuracy and user identification. The suggested framework includes three entities sender, deliverer, and customer. The system uses a one-time QR code to confirm the meeting of entities. Once scanned, it triggers a function that verifies and commits a block to the chain. The sender initiates the smart contract to create the shipping request, which is broadcasted to the available deliverers. Once one of the deliverers accepts the request, he meets with the seller and confirms package handover by scanning the QR code generated by the sender. When the deliverer arrives at the customer location, he uses one time QR to record the parcel delivery. Although their approach automates the delivery process, how it verifies identities is ambiguous. Order Cancellation is not handled as well by their scheme. Besides, there is no incentive for entities to behave honestly while a third party is absent.

Demir et al. in [19] deployed a framework for delivery assurance called BIDAS that addresses two main issues in the delivery sector: parcel handover and continuous monitoring. BIDAS suggest a blockchain-based solution integrated with the Internet of Things (IoT), such as sensors, to track handovers. Participants' devices can communicate if this is the correct destination to prevent incorrect deliveries and lost packages. The devices can interact through NFC and read the RFID of the parcels to mark the parcel as delivered. Regarding continuous monitoring, BIDAS allow buyers to provide their opinions about the experience and share them with all participants. The research lacks an implementation of the framework's conceptual data model and activities of the e-commerce delivery system. Hence, the system performance in terms of cost, scalability and privacy cannot be determined.

Wang et al. in [88] presented an auditable protocol for transparent, tamperproof, and verifiable transactions between three entities, merchant, logistics company, and consumer. The schema requires a third party, called a regulator, responsible for au-

thenticating users interested in participating in the network, and registering a smart contract with their data in the Blockchain. It has offered a pre-verification technique to prevent replacing products during delivery by the courier. Also, it discussed the return of the product in two cases; when consumers receive their products and when the delivery time of the products exceeds a predefined period. Similar to previous works, it require secure deposit per order equals to parcel worth. Depositing a collateral approach doesn't fit the crowdshipping environment. To provide a practical and cost-effective solution, it would be unfeasible for the courier to deposit collateral that equals each package value he will ship or the double. Because the courier will have to deliver several parcels on the same route. It is estimated that a delivery driver typically delivers 40–70 packages each day for close allocation of addresses and 125–200 packages if the addresses are super clustered [87].

Ha et al. in [35] proposed a novel ACL cash-of-delivery solution to address the courier's traceability during the delivery process and to integrate access control protocols to protect sellers' and customers' privacy. A hash code is created for each package based on its details and verified to ensure no change happens to the order details. Smart contracts include a penalty mechanism when trouble occurs, such as a damaged, missing, or incorrect package. The customer selects the shipper. Once agreed, the system sends the hash and the seller's contact to the shipper. The system verifies identity and hash between seller-shipper, shipper-shipper, and shipper-customer. Only the shipper mortgage a deposit equal to the package value, which is transferred to the seller in case of late delivery or the customer rejects the item if it was changed. When the customer receives the package, the shipper takes the cash payment and divides the profit with the seller.

3.3 Reputation Based Systems

A wide variety of trust management strategies have been presented in diverse environments, such as peer-to-peer, e-commerce, and multi-agent. This section focuses on the most significant ones we used to create our suggested reputation system.

EigenTrust is the most known and widely used reputation algorithm in a P2P network [18] which recommends a method to aggregate the local trust values of all peers (direct evaluations of a transaction). In EigenTrust, besides the local trust values, each peer is assigned a unique global trust value to represent the interactions with all other peers. Peers can rate another peer as either positive or negative -1, +1. To measure the global reputation score, one must normalize the local trust values first and then aggregate the normalized values. Since each peer keeps a list of all peers who have interacted with before, when one peer comes across a stranger peer, he can ask other credible peers for feedback on the stranger's trustworthiness. A notable disadvantage of EigenTrust is that it doesn't distinguish between unsatisfactory experience and no interaction, which makes it vulnerable to traitor attacks.

Another feedback-based trust model for P2P network environments is **PeerTrust** [89]. Li Xiong and Ling Liu identified five criteria to assess a peer's reputation, feedback received from other peers, the total number of transactions a peer performs, and the credibility of the feedback source. Plus, a transaction context factor in distinguishing between mission-critical and less-or non-critical transactions and the community context factor to address community issues. The authors aggregate the five criteria into a general trust metric and develop a corresponding formula to determine a peer's trust value, where the parameters are weighted appropriately. PeerTrust employs an adaptive time-window-based computation method to avoid Traitor attacks by ensuring reputation will not increase or decrease within an unrealistic threshold. Furthermore, it leverages a PKI-based scheme and data replication to enhance the security and reliability of the trust data management.

Sidra Malik et al. in [53] argue that blockchain technology alone cannot guarantee the confidence and reliability of data regarding the quality of commodities and the trustworthiness of supply chain participants. Because once published on the Blockchain, false data produced by supply chain entities becomes immutable. Therefore, to improve the trust and reliability of the data, their research propose a three-layered trust management framework called **TrustChain** consisting of data, blockchain, and application layers to overcome trust-related problems in the supply

chain industry. The reputation framework leverages smart contracts for transparent, effective, secure, and automated computation of reputation scores. Raw data are stored off the chain and include sensor data streams installed to observe temperature, location, and humidity, trade events such as change of ownership, quality assessment conditions specified in the smart contract, and regulatory endorsement data. At the blockchain layer, transactions are stored on the ledger and governed by ACL. The application layer address queries and transaction requests from different supply chain participants. Their schema employs a forgetting factor to give recent events higher weights than older events. As a reward, it publishes the entities with the highest trust values on the network. As a penalty, it revokes the entities from participating in the network for a certain period of time.

Zhou et al. in [92] propose a blockchain-based decentralized reputation system (BC-DRS) in the E-commerce platforms. The reputation scores of participants are computed using three factors, transaction time, transaction amount, and entity history of the user. Then the user's reputation score is calculated through the positive ratings ratio of all the ratings of the user's transactions. The smart contract verifies whether a transaction has happened or not. And if both the buyer and the seller have submitted the comments and the ratings. To encourage honest comments and ratings, a certain amount of monetary compensation is sent from the seller to the buyer after each transaction. The proposed BC-DRS system is deployed on Ethereum. Their experiment result revealed that the fee costs of the model are minimal, approximately 0.1 USD. Moreover, the result showed that their blockchain-based model helps to resist several common attacks, i.e., unfair rating and colluding, since malicious entities need to make transactions with a lot of time and money to such attacks. There are two limitations to this work. First, the reputation score becomes constant after a particular number of transactions. Second, the reward gradually increases until it reaches more than 400$, which doesn't seem feasible in production e-commerce platforms.

Truong et al. in [84] built a universal decentralized framework that works alongside any DApp to assess trust relationships between ecosystem members. This trust

system functions as middleware between a BC platform and decentralized applications, providing end users with the capability to establish and maintain a network of trust connections. Additionally, the study aims to provide a system that successfully defends against attacks on reputation (such as Sybil, whitewashing, self-promotion, and bad mouthing). The study solely uses two factors; reputation and experience. Experience expresses the feedback one party offers another after a transaction. The reputations is the outcome of all aggregated experiences which are determined following the weighted Google PageRank algorithms. Finally, trust between two entities is computed as a combination of reputation and experience. The study developed a proof-of-concept system implementing the trust model on top of the Ethereum public blockchain. The system latency results show a technical limitation on the performance of the Ethereum-based system, which restricts the effectiveness of the proposed decentralized trust system to only small-scale services. Another drawback of the suggested strategy is that the reputation computation is conducted off-chain using a technology called Oraclize since reputation smart contracts are pretty expensive. This approach could result in a reliance on a third party (Oraclize provider), which would go against the idea of a decentralized system.

CHAPTER 4

Methodology

This chapter describes the adopted threat modeling framework and obtained attacks list and potential countermeasures. Besides, it introduces several reputation management system threats and the proposed reputation model formation and parameters.

4.1 Threat Modelling

Blockchain technology has the role of recording system data and maintaining the recorded data's integrity in a decentralized system environment. Researchers and practitioners have found that despite blockchains' attempts to deter attackers by replicating the database and code execution, there is still no shortage of vulnerabilities. The best practice for developing a secure and reliable system is to conduct a threat modeling stage to investigate potential security vulnerabilities. Threat modeling is a common practice conducted by organizations to analyze system architecture and its security objectives and identify potential threats and the associated mitigation controls. Such a model can direct programmers in implementing the proper defenses during the design stage and evaluating the system's security after design.

Different methods to organize threats have been proposed in the literature. Some well-known threat modelling methods are STRIDE, VAST, PASTA, Trike, Attack tree, and Octave [70]. In this study, we will use the STRIDE methodology due to its simplicity and maturity. Microsoft developed the STRIDE approach in 1999 as one of the earliest works in the threat modeling field. STRIDE is one of the most used strategies in threat modeling. It is an acronym for threat categories considered

in the framework: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [71]. Table 4.1.1 shows the STRIDE threats, a definition, and the corresponding property that system need to maintain . STRIDE consists of three steps :

1. **Architecture modeling**: the system under analysis is decomposed and modelled using Data Flow Diagram (DFD) that represents how information moves around in a software-based system using processes, external entities, data storage points, and trust boundaries.

2. **Threat category mapping**: Each DFD element is examined in-depth and mapped to one of the potentially applicable STRIDE threat categories.

3. **Threat elicitation**: This step utilizes a checklist-based approach to derive concrete threat cases based on the identified mappings.

Threat	**Threat Definition**	**Property Violated**
Spoofing	Pretending to be something or someone other than yourself	Authenticity
Tampering	Unauthorized modification of data on transit or at rest	Integrity
Repudiation	Claiming that you did not do something (honestly or deceptively)	Non-Repudiation
Information disclosure	Gaining access and exposing information to unauthorized persons	Confidentiality
Denial of service	Preventing system from providing service by absorbing resources needed to provide service	Availability
Elevation of privilege	Allowing a program or user to do something they are not allowed to do	Authorization

Table 4.1.1: STRIDE threat definition and property violated

To organize the attacks of the proposed LMD system, we identify three security domains based on the system-adopted technologies. A Blockchain domain contains the threats of blockchain elements such as network, user wallet, smart contracts, and consensus. A user-centered domain includes the behaviour of the user in the system and associated threats. A Crowdshipping domain handles risks related to package delivery systems using the crowd. Application domain comprises the mobile/web application components threats related to a web server, API, and database servers will be out of the scope of this paper.

4.1.1 Blockchain Domain Threats

The proposed LMD platform targets permissioned Blockchain. To gain an overview of relevant attacks, we performed a literature review of attacks on blockchain systems. Then we extracted threats applicable to private blockchains from the surveyed papers. This section highlights the blockchain-specific risks, broken down into categories based on the targeted component.

4.1.1.1 Network Threats

A blockchain network includes nodes that create and broadcast transactions and nodes that provide other services such as transaction time stamping, hosting smart contracts, validating the transactions, and committing the blocks. Attackers look for network vulnerabilities and exploit them to isolate these nodes from the network, restrict their access to the network's resources, or create a partition in the network and enforce conflicting rules among the peers with the following types of attacks.

- **Distributed Denial of Service (DDoS)**

  Although Blockchain technology is a distributed peer-to-peer system, it is still prone to DDoS attacks. DDoS on a Blockchain system refers to an attacker attempting to cause a complete or partial service disruption by consuming system processing resources with a tremendous amount of requests in a short time. DDoS could target the memory and storage resources, reducing the peering

and consensus capabilities. For instance, an inside attacker can overwhelm the network with valid transactions, which may lead to memory pool depletion, resulting in a system crash [39]. However, this attack is economically expensive as the attacker has to pay a transaction fee for each. Besides, an adversary can launch a DDoS attack if he controls enough voting power (e.g., third of total replicas in BPFT) by compromising them [17]. When the primary node sends the transaction for verification, compromised replicas will not reply with their approval. Hence, the system's operation will be halted due to the inability to process transactions.

Countermeasures: Collecting performance metrics of the blockchain network, such as transaction throughput and latency, help in the early detection of DDoS attack that targets system availability. The use of Access Control List (ACL) which permit only allowed packets and deny all others based on predefined rules help to stop these attacks in early stage.

- **Eclipse Attack**

  Eclipse or Netsplit attack aims to isolate honest nodes in the network by controlling their incoming and outgoing traffic and feeding them with fabricated information regarding blockchain and transactions. Attackers hijack the node's connections using connection monopolization or poisoning routing tables. This attack might target controlling pieces of network communication and divide the network to increase synchronization delay or deceive it with a full fake version of the blockchain. The authors in [54] performed a low-cost eclipse attack that exploits the Kademlia peer discovery logic of Go Ethereum using only two machines.

  Countermeasures:If the blockchain relies only on ECDSA public keys as the sole node identifier, it is recommended to use a combination of IP address and public key instead to mitigate eclipse attack [80]. Machine learning as well detects Eclipse attacks by monitoring the network using a random forest classification algorithm. In [90] the authors used a collection of regular data packets

and attack data packets to train the model, which achieved 71% precision and 95% recall of detection. However, this approach is dataset-dependent and challenging to implement when data is unavailable. Another method relies on the suspicious block timestamp. If the time between the current block and the previous block is long, the network has been partitioned. The side effect of this solution is the speed, as it may detect the attack within 3 hours [2].

- **DNS Attacks**

  When a node joins the BC network for the first time, it is unaware of the active peers. DNS acts like a bootstrapping approach to discover and obtain further information about other active peers in the network. DNS opens various attack surfaces to the blockchain networks in general. For example, a DNS spoofing attack, also known as a DNS cache poisoning attack, where the adversary poisons the DNS cache at the resolver and force the server to return a false value. When a user queries the server to obtain the IP addresses of peers who accept connections, the user is routed to the attacker's network. Then attacker deceives the user with fake blocks and transactions [68]. In DNS Hijacking, the adversary shuts down DNS; consequently, all blockchain network users will not be able to validate their authorization and certificates due to the inaccessibility of the identity provider.

  Countermeasures: Implement best practices such as logging and monitoring inbound and outbound queries to detect anomalies. Additionally, DNS Security Extensions such as DNSSEC [79] protect servers from undesirable access and tampering by controlling and authenticating DNS queries and responses using digital signatures based on cryptography. Or enable multi-factor authentication on the domain registrar account and use a registrar lock service to request permission before changing DNS records.

4.1.1.2 Authentication and Access Control Attacks

This section deals with the most prominent attacks if some participating entities in the blockchain network are compromised.

- **Identity Provider Compromise**

  The centralized aspect of permissioned blockchain lies in the identity provider and the associated Certificate Authority (CA). When the identity provider is malicious/compromised because of private key theft, the attacker can handle the identity provider's administrative controls, such as adding and revoking identities to and from the network and the level of the given access to the nodes. A compromised identity provider can cause catastrophic damage to the network since it is the sole authority that manages supported identities and generated keys used to sign transactions. The following attacks could occur as a result of compromised identity provider:

  1. **Sybil Attack**

     In our platform, each node must prove its identity before joining the network; hence, it will not be able to forge identities. However, identity provider contains a list of permissioned identities, making it a single point of centralization. Suppose the identity provider is compromised due to a rogue insider or private key theft. In that case, an attacker could flood the network with fake identities and use the majority to cause harm to the system.

  2. **Blacklisting Attack**

     Identity provider has some parameters specified to allow for identity validation. One of them is the certificate revocation list (CRL), which consists of certificate serial numbers of certificates that are no longer valid [42]. Assume that an inside attacker gains control of the identity provider. He can manipulate the access level of identity, add or revoke access, and blacklist specific identities, consequently causing blockchain services disruption by

blocking valid transactions.

Countermeasures: From the blockchain framework perspective, the theft of private keys cannot be detected. However, dishonest behaviour using these keys can be discovered [65]. Logging identity provider actions, such as certificate creation and revocation, can help find malicious behavior in case of compromise. Besides, alerting based on that logging results in early identification and remediation. Moreover, An Intel Software Guard Extension(SGX) proposed in [50]to secure the identity provider during member registration, enrollment, transaction signing, and verifying. Intel SGX is a trusted execution environment for applications that isolates a portion of physical memory (an enclave) to protect code and data from unauthorized access or modifications. Another mitigation action of sybil attack is requiring that an individual IP address cannot create too many user accounts in a given time interval [57].

- **Certificate Authority Attack**

  If the LMD blockchain network relies on only a single root Certificate Authority, then it is a single point of failure, and if it gets compromised, the entire network is at risk. The attacker will be able to generate unauthorized certificates and cause a detrimental effect on the security of the identity provider by mounting effective large-scale man-in-the-middle attacks.

  Countermeasures: A regular audit of trusted certificate authorities using log-based PKI extensions is necessary to identify unauthorized changes to the environment that may allow certificates to be issued to unauthorized users. Distributed trust PKI architecture could alleviate the impact of the CA security breach, as it will only affect its certified end users. Furthermore, [12] proposed a decentralized PKI transparency that allows participating entities to record any valid certificate or revocation that is presented to them in order to provide a global audit.

- **Quantum Attack**

  The computational complexity of cryptographic techniques adopted in blockchain assures its security. Asymmetric encryption schemes such as RSA or Elliptic Curve Cryptography (ECC) are used to generate private/public key pairs. However, the beginning of quantum computing evolution helps the Grover algorithm generate hashes, and the Shore algorithm searches discrete logarithms and factoring integers more quickly. Thus, threatening public-key cryptography and hash functions. In [86],[1], researchers concluded that in 2035 it is more likely that a quantum computer operating at 10 Mhz would be able to break RSA 2048 cipher in roughly 42 min. If the attacker used Shore algorithm to break the cryptographic algorithm and find the private key using the user's public key, he will invade the privacy of LMD platform users, have unlimited access to the blockchain, and perform unwanted transactions.

  Countermeasures:Numerous studies in the literature proposed two ways to mitigate quantum attacks. The quantum-resistant public-key cryptographic algorithms, such as the lattice-based signature scheme proposed in [28] and [48]. Besides, the Quantum-secured Blockchain [44] that offers unconditional security by leveraging Quantum Key Distribution (QKD). QKD can generate a secret key between two parties connected by a quantum channel and a public classical channel. However, most scientists agree that this is unlikely to happen in decades due to its complexity and high cost.

- **Wallet Theft**

  LMD platform users' private keys must remain undisclosed to the public since it allows them to access the blockchain network and verifies transactions, and it is produced once and cannot be recovered if lost. Malicious actors employ various tactics to steal private keys such as phishing, dictionary attacks [23], or prominent crimeware families such as Atmos, Dridex, Ramnit, IceID, and many others [22].

  Countermeasures: The private keys used by nodes should be physically pro-

tected, using technology such as hardware security modules (HSMs). HSMs refers to hardware security module (HSM) is a physical computing device that safeguards and manages digital keys, performs encryption and decryption functions for digital signatures, strong authentication, and other cryptographic functions. The use of HSMs ensures that the private keys cannot be read from server memory if a node is compromised [41].

4.1.1.3 Consensus Attacks

This section examines the attacks related to consensus and transaction validation. Transactions verification takes a certain amount of time, which creates a perfect vector for cyber attacks. We introduce only threats related to CFT and PBFT consensus algorithms. Any of them is better candidates for consensus in private blockchain, since all participants are whitelisted and constrained by strict obligations to conduct appropriately [77].

- **Consensus Delay Attack**

  A malicious validator can propagate invalid blocks, resulting in a consensus delay as the other peers waste computing power on verifying invalid blocks.

  Countermeasures: Monitoring round-trip time (RTT) and the number of discarded blocks received between validators is vital to detecting such attacks.

- **Compromised Primary Replica**

  PBFT assumes that the primary node which orders transactions and obtains approvals from other replicas is trustworthy. This assumption may lead to vulnerabilities in the private blockchains. A compromised primary replica in PBFT could perform several attacks. For example, rearranging the transaction order to cause block verification delay, falsifying transactions even after receiving other replicas' approval, ignoring correct approvals from other nodes, and early halt the transaction execution. Additionally, it can launch a block withholding attack after receiving the confirmations from other nodes [68].

Countermeasures: Since all the network participants know the identity of the primary node, malicious behaviour of a primary can be tracked back, eventually.

4.1.1.4 Smart contract Attacks

The execution of LMD operations is defined by the smart contracts that all the participating parties have acknowledged. The development of smart contracts can be prone to several programming errors and hidden vulnerabilities that can ultimately lead to accuracy and security violation, bugs, and faulty behavior. We present some of these vulnerabilities below:

- **Vulnerability Injection**

  A malicious insider may intentionally inject a vulnerability into the smart contract. Once the operational software is upgraded, the network operators deploy the vulnerable version of the smart contract into production, permitting the attacker to abuse the vulnerability. Some smart contracts' languages rely on packages imported from public version control sites like GitHub. These packages might suffer from programming defects, which leads to an unpredictable state of the smart contract [72]. The insider may exploit these defects to modify the state of smart contracts in his favor.

- **Source Code Vulnerabilities**

  Coding errors and design flaws are the main reasons that cause the smart contracts' vulnerabilities that lead to a series of attacks and possible exploits.In [51] the study showed that 45% of Ethereum smart contracts are vulnerable to several bugs such as the following listed bugs.These bugs may manipulate the contract outcome state, discourage consensus, and compromise privacy by sending confidential information to unauthorized parties. Additionally, users may be able to execute unauthorized operations due to permission management breakage in the smart contract [66].

  1. **Reentrancy** is one of the most destructive attack techniques in the smart

contract [69] as it may destroy the contract or steal valuable information. Reentrancy occurs when a function calls for another smart contract through an outer call.

2. **Error Handling Exception** Mishandled exceptions may appear when different smart contracts are called from each other. When contract X calls contract Y, Y contract will stop executing and return a false result if it runs abnormally.

3. **Timestamp Dependency** occurs when the smart contract uses timestamps in its execution. An entity could change time to manipulate code execution and output to his own benefit.

Countermeasures: Smart contracts developers should follow a secure Software Development Life Cycle Framework in the design phase to safeguard the system from security flaws. Before deployment, smart contract security should be assessed with smart contract analysis tools, an external security audit, or formal verification.

4.1.2 Crowdshipping Domain Threats

This section describes the threats that come to the scene in any parcel shipping platform.

- **Warshipping**

  The attacker uses an inexpensive and low-power device to perform close-proximity attacks remotely from any location. Attackers hide a tiny device (similar to the size of a small cell phone) in the package and ship it off to their victim to gain access to their network. The device has a 3G-enabled modem and a wireless chip; it costs around $100 to build. The device conducts periodic scans for nearby networks to track the parcel's location. When the package arrives at the target destination, the attacker would be able to control the network remotely and start to attack the target's wireless access. Thus, he can discover

weak spots, monitor the system communication, and steal sensitive data or user credentials. [38].

Countermeasures: It is vital to inspect the received package and dispose of any packing materials. Suppose enterprises would utilize the LMD platform for their deliveries. In that case, it is recommended to employ a multi-factor authentication to access WiFi or secure the network with an intrusion detection system (IDS), a device or software application that monitors a network for malicious activity or policy violations [8].

- **GPS Spoofing Attack**

  Malicious couriers are using GPS spoofing apps on their mobile devices to fake their location and deceive the LMD platform into believing their position to be somewhere else than where it is or to be located where it is but at a different time. The reasons behind that are to be able to receive shipping requests for an area they don't have access to based on the actual location or to get paid without actually delivering the package at all [56].

  Countermeasures: There are many existing countermeasures to detect GPS spoofing. For instance, [61] proposed a novel approach, validating the GPS position by comparing it with the position obtained by the Base Station (BS) that belongs to the mobile cellular network infrastructure. This method is effective in cellphones and other devices with a cellular network.

- **Courier Impersonation**

  An attacker may steal a victim's identity, such as a driving license and Social Security number, and set up an account on the LMD platform. He could exploit the fake account for their purposes and act maliciously or sell it to someone who is illegible to do shipping tasks [34].

  Countermeasures: LMD platform requires users to reveal their identities. The LMD platform may verify the identity of a user by using the dual-process method to avoid impersonation or faked identities. Users can provide proof of

two documents from reliable resources, for example, a government-issued ID like a driving license or passport. However, to assure that the IDs are authentic, the LMD platform may ask the user to scan them using the camera on their mobile phone or electronic device and utilize a technology to compare the features of the provided IDs against known characteristics [25]. Alternatively, users can use Digital ID, an electronic version of trusted government identification protected by strong encryption to offer adequate safety, better security, and more robust privacy than physical identification documents [62].

- **Package Manipulation**

  A courier could manipulate the package in various way, such as intentional damage, theft or replacement. He can act maliciously and change the package then claims it is the same one received from the shipper.

  Countermeasures: The handover of the package between the supplier and the courier should verify the package attribute such as its appearance picture, parameters, and value.

- **Illegal Packages**

  The LMD platform's users can exploit the system to send hazardous materials, dangerous items, or illegal products that are not permitted by law to be transferred. **Countermeasures**: It is the shipper's responsibility to comply with current government regulations or laws. Therefore, it is essential to obtain consent from the shipper that the package does not violate the laws.

4.1.3 Attack Vector

Attack/Category	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevated Privileges
DDoS					✓	
Eclipse Attack		✓			✓	✓
DNS Attack		✓			✓	✓
Identity Provider Compromise						✓
Certificate Authority Attack	✓					✓
Quantum Attack		✓		✓		✓
Wallet Theft	✓	✓	✓			
Consensus Delay		✓			✓	
Compromised Primary Replica		✓			✓	✓
Vulnerability Injection		✓				✓
Source Code Vulnerabilities		✓		✓		✓
Warshipping				✓		
GPS Spoofing Attack	✓		✓			
Courier Impersonation	✓					
Package Manipulation			✓			
Illegal Packages						✓

Table 4.1.2: Mapping between LMD platform potential security risks to STRIDE threat categories

4.2 Reputation Management System

According to Gambetta et al. [27], Trust is the subjective probability by which an agent evaluates that other agents will undertake a specific action that is beneficial or at least not damaging. Reputation is an assessment of a person's standing or character and is crucial in determining the trustworthiness of a particular entity. A key distinction between reputation and trust is that the latter uses an objective standard (such as the history of interactions outcomes) as input. In contrast, the former uses a subjective assessment. This difference can help determine an entity's reliability or trustworthiness.

Centralized reputation systems (CRSs): One efficient way to manage a system's reputation is to appoint a centralized authority server as a trust representative. Central servers process and store the ratings. Many well-known online retailers like eBay and Amazon have established their own CRSs. Trust management of a system's participants is made relatively simple by the centralized servers, which may alleviate many trust concerns to a certain extent. However, the core of mistrust in the CRSs cannot be fundamentally addressed for the following reason. The centralized authority server is susceptible to errors, manipulation, and collusion. As a result, it is vulnerable to several harmful actions that unfairly harm the participant's reputation. For example, several drivers in DoorDash claim that they experience a lot of customer rating manipulation [82] . Additionally, there is a shortage of efficient incentive mechanisms, contributing to the abundance of default comments and ratings in the CRSs. These provide customers with minimal guidance for making transaction decisions.

DRSs (Decentralized/Distributed Reputation Systems): These types of systems are developed for P2P networks to locate reliable resources, encourage honest conduct from participants, and evaluate the quality of provided service or content. Numerous DRSs for P2P networks have been proposed in the literature. These DRSs are divided into systems based on local reputation, systems based on global reputation, and systems that combine local and global reputation. Although some of the well-known issues with reputation systems have been solved, DRSs in P2P networks still have a

number of obstacles. First, compared to CRSs, reputation systems will increase the P2P networks' computational burden. Second, it is highly challenging to maintain the reputation data's accuracy and quickly disseminate it to a considerable number of dynamic peers. Last but not least, the DRSs of P2P networks are ineffective against attacks like reputation alteration and common ones like collusion and unfair rating.

Since reputation management systems' primary goal is to increase trust among members of online communities, a wide range of fields utilizes these systems. In E-commerce websites such as eBay, Amazon.com, and Etsy. Online advice communities such as Stack Exchange. Content rating and discussion platforms such as Reddit. Programming communities such as Stack Overflow and many others.

4.2.1 Threat Scenarios in Reputation Systems

A security mechanism is a procedure or a device created to identify, stop, or recover from a security attack. Unlike several security mechanisms like access control and authentication that enable or refuse a user's access to a resource, reputation systems do not offer a way to stop or detect a security attack directly. Reputation system outline a procedure to spot malicious individuals and prevent them from harming other participants. Reputation systems must constantly gather feedback about the behavior of their users to have an accurate record of their reputation. It offers a means to determine whether a transaction between interacting parties has a high chance of success or a low probability of failure. However, Several classes of attack can exist in trust and reputation systems, as mentioned in [45] are:

- **Whitewashing attack**: Whitewashing is the process of exiting the system with an account with a poor reputation and rejoining the system with a new identity to obtain the initial trust score again.
- **Slandering attack**: Slandering or Bad Mouthing describes an attacker (or groups) who creates false negative feedback about other identities to damage their reputation.

- **Sybil attack**: Sybil attack refers to a scenario in which a single malevolent peer creates numerous identities and employs them cooperatively to subvert the system.

- **Ballot Stuffing**: Ballot Stuffing is when a dishonest peer tries to generate false transactions with himself or colludes with other peers to boost his reputation.

- **Impersonation**: Impersonation attack involves a malicious peer impersonating another peer. The attacker can then act dishonestly on behalf of the impersonated peer, negatively impacting that peer's reputation or propagating false ratings about others using the stolen identity.

- **Repudiation of transactions** In this attack an entity may deny that it has sent or received a rating. By not being able to verify that a participant is responsible for such actions, malicious peers have no fear of being identified and penalized.

- **Traitor Attacks** In this scenario, a peer behaves honestly for an initial period of time to build up a positive reputation before deceiving other users or starting a whitewashing attack.

4.2.2 Proposed Reputation Model

Reputation plays a crucial role in establishing systemic trust since each participating entity has a reputation value that represents the reliability of the provided service and behavior. Previous works in chapter 3 established trust by reserving collateral deposits from each entity in the form of cryptocurrency. Instead, this work construct trust through a reputation-based network that uses blockchain immutability and offers a confidence reputation score in an environment that lacks a third party who maintains transparency and solves disputes between participants. Each courier is assigned a reputation score that sellers will consider during the selection process. A high reputation score reflects the good courier behaviour. Furthermore, unlike traditional crowdshipping schemes where the reputation is managed and controlled by a

third party, our reputation system is completely decentralized and implemented on blockchain.

4.2.2.1 Local Reputation

It is not fair to directly use users evaluations to determine the reputation scores of the courier. Multiple transactional elements such as shipping order time, shipping cost, the credibility of the evaluation's source, and the courier's completed orders are disregarded in the reputation's computation, making the reputation system vulnerable to attacks. To calculate the courier's reputation score, we weigh the received evaluation, denoted by r and take value of [-1,1], according to the following factors:

1. **Order time**: If the courier accepts several shipping orders in a short time, it is possible that the courier and the seller collude to receive good ratings. Thus, for a current order, if the courier's previous order occurred a short time ago, the rating of the current order should be set as a relatively low value to deter the collusion attacks. Thus, for the rating, the weighting factor of order time is defined using the Tangent hyperbolic function:

$$\varphi(\triangle T) = tanh(\triangle T) \tag{1}$$

   where, $\triangle T$ refers to the time interval between the timestamp of current order T and that of previous transaction T' divided by T_a which refers to the normal frequency of user interaction in the system, $\triangle T > 0$ and $\varphi(\triangle T)$ ranges from 0 to 1. Larger time interval $\triangle T$ will lead to higher value of the weighting factor.

2. **Shipping Cost**: It is not costly for the seller to create orders with low asked prices to boost courier ratings or unfairly submit low ratings to the courier. Therefore, to overcome this concern, the rating of an order should be related to the shipping cost amount. The weighting factor of the shipping cost amount is defined by the following equation, where V refers to the shipping cost and $\psi(V) < 1$. The shipping cost is divided by 10 to avoid convergence to 1 very

quickly:

$$\psi(V) = 1 - \frac{1}{(1 + \frac{V}{10})} \tag{2}$$

3. **Number of shipping orders**: A courier may increase his trust value by being active in the network and increasing the fulfilled shipping orders. To distinguish a courier who has a high reputation for a few good orders from a courier who has a high reputation but with a large volume of orders, we need to consider the number of transactions. This factor is denoted by TX_N and defined by considering the maximum number of completed orders by couriers in the network and assigning different weight values for different ranges. The network determines what factor's score to assign for each range of shipping requests and update it periodically. As an illustrative example and assuming that the maximum number of orders in the network is 40, table 4.2.1 show a mapping between the number of shipping requests with the corresponding factor's value.

Number of Completed Orders	**Factor Value**
0 - 10	0.25
11 - 20	0.50
21 - 30	0.75
31 - 40	1

Table 4.2.1: Example of Number of Shipping Orders Factor Value

4. **Credibility of the Local Rating**: The proposed reputation model must be robust against unfair ratings. A participant may make false statements about the courier's service due to jealousy or other types of malicious motives. Consequently, a trustworthy courier may end up getting a large number of unsatisfactory ratings even though it provides satisfactory service in every order. Therefore, we consider the fairness of the provided rating score for each order to protect the courier from such incidents. At first, we calculate the average of all ratings given to a particular courier, say c_j. The equation 3 shows how the

average is calculated. r_{kj} refers to rating received by user u_k toward the courier c_j. The N_j refers to the total number of fulfilled shipping requests by courier c_j .

$$\overline{R_j} = \frac{\sum_{k \in N_j} r_{kj}}{N_j} \tag{3}$$

In the second step, we calculate the average of all ratings given to courier c_j by the rater, say u_i using the equation 4. The N_{ij} refers to the number of ratings that u_i has provided toward the courier c_j.

$$\overline{R_{ij}} = \frac{\sum r_{ij}}{N_{ij}} \tag{4}$$

In the third step, we calculate the standard deviation of all ratings given to a courier c_j as we show in equation 5.

$$SD_j = \frac{\sqrt{\sum_{k \in N_j} (r_{kj} - \overline{R_j})^2}}{N_j} \tag{5}$$

Finally, we define and calculate rating credibility of relations between raters and couriers regarding the equations 3,4,5.

The credibility of rating provided by r_i toward c_j shows how fairly r_i has evaluated c_j and is denoted by Cr_ij . The Cr_ij is calculated as follow:

$$Cr_ij = \begin{cases} \frac{\overline{R_j} - SD_j - \overline{R_{ij}}}{Max_{Rep}} & \text{if} \quad \overline{R_{ij}} < (\overline{R_j} - SD_j) \\ 1 & \text{if} \quad (\overline{R_j} - SD_j) \leq \overline{R_{ij}} \leq (\overline{R_j} + SD_j) \\ \frac{\overline{R_{ij}} - (\overline{R_j} + SD_j)}{Max_{Rep}} & \text{if} \quad (\overline{R_j} + SD_j) < \overline{R_{ij}} \end{cases} \tag{6}$$

where Max_{Rep} refers to the maximum value of the reputation score and equal 1.

According to Equation 6, the averages falling in $\overline{R_j} \pm SD_j$ are trustworthy and dependable, but those that fall out of that range have very low credibility,

and their impact on reputation are decreased. The Cr_j shows how close is the judgment of u_i to the majority consensus about courier c_j's trustworthiness. Thus, we use credibility to reduce the effect of ratings provided by raters who disagree with the majority consensus about the courier. It is worth noting that the credibility factor could be difficult to measure if both rater and the courier are transacting for the first time. Therefore, the credibility factor value will be 0.5 as the probability of a fair/unfair rating score is 50%.

Next, for each shipping request we compute the local reputation score of a courier by the above weighting factors. Local reputation score ranges from 0 to 1 . We weigh received rating r (positive or negative) by the following equation:

$$e = \frac{(r \times Cr) + \psi(V) + \varphi(\triangle T) + TX_N}{4} \tag{7}$$

4.2.2.2 Global Reputation

As mentioned before, each courier has a local and global reputation score. When the courier joins the network, an initial global reputation score is assigned, equaling 0.5. The global reputation is an indicator of trust and increases or decreases once a new local reputation score is added. Also, the global trust score decays if the courier has not been active and has taken shipping requests after a period of time. The local reputation score e could be satisfactory or unsatisfactory depending on a predefined threshold θ. The amount of increase, decrease, and decay depends on the local reputation score e and the current value of the global reputation $GRep$, which can be modeled by linear difference equations and a decay function as follow:

Increase model The current global reputation score denoted by $GRep$ increases once updated with a satisfactory transaction (at the time t, indicated by the local reputation score $e_t \geq \theta$) that follows the linear difference equation:

$$GRep_t = GRep_{t-1} + e_t \times \Delta GRep_t \tag{8}$$

where $\Delta GRep_t = \alpha \times (1 - \frac{GRep_{t-1}}{Max_{Rep}})$ and α is the maximum increase value of global

reputation score in two consecutive shipping requests and Max_{Rep} is the maximum global reputation and equals 1.

Decrease model Similarly, $GRep$ decreases if the local rating is unsatisfactory (indicated by the local reputation score $e_t \leq \theta$), following the equation:

$$GRep_t = Max(Min_{Rep}, GRep_{t-1} - \beta \times (1 - e_t) \times \Delta GRep_t) \tag{9}$$

The Min_{Rep} is the minimum global reputation and equal 0. The decrease rate $\beta > 1$ implies that it is easier to lose the global reputation value due to unsatisfactory rating than to gain it (by a satisfactory rating).

Decay model Global reputation decays if there is no transaction after a period of time and the decay rate is assumed to be inversely proportional to the strength of constructed trust relationship of the courier (value of the Reputation). Based on these observations, the Decay model is proposed as follows:

$$GRep_t = Max(Min_{Rep}, GRep_{t-1} - \Delta Decay_t) \tag{10}$$

where $\Delta Decay_t = \delta \times (1 + \gamma - \frac{GRep_{t-2}}{Max_{Rep}})$, and δ is the minimum decay value ensuring any global reputation degenerates if it is not maintained. And γ is a decay rate controlling the amount of the decay. The network administrator can identify the inactivity threshold (e.g., three months) and periodically compare the network's couriers' last shipping request. If the courier has been inactive more than the threshold, his global reputation will be declined using equation 10, and vice versa.

The seller might delegate the shipping requests to a courier with a particular reputation score in our proposed platform. The system will verify that the candidate couriers match the minimum reputation threshold specified by the seller. Once the request is completed, the smart contract calculates the weighted local reputation score and updates the courier's global reputation; more details is discussed in chapter 5.

4.2.2.3 Example

The following table 4.2.2 represents data required to compute the local and global reputation score for Courier c_j. We assume that the courier completed 23 positive ratings, and 2 negative ratings. Courier c_j completed a shipping request for Seller u_j previously at the indicated time in the table, in both times Courier c_j received a positive ratings. So, the average ratings $\overline{R_{ij}}$ from Seller u_i to Courier c_j is 1. Now we

Courier's Global Reputation $GRep_{t-1}$	0.88
Previous Shipping Request Timestamp T'	31/12/2022 12:35:00
Current Shipping Request Timestamp T	31/12/2022 13:10:00
Normal interaction Frequency T_a	60 minutes
Shipping Cost V	30$
Courier Total shipping Requests	25
Courier's Ratings Average $\overline{R_j}$	0.84
Courier's Ratings STD SD_j	0.54
Local Reputation Threshold θ	0.5
Increase Rate α	0.1

Table 4.2.2: Reputation Example data

calculates the local reputation factors, table 4.2.3 presents these calculation.

Factor	**Equation**	**Outcome**
Order Time	$\triangle T = \frac{(13:10:00 - 12:35:00)}{60} = 0.583$ $\varphi(0.583) = tanh(0.583)$	0.525
Package Value	$\psi(30) = 1 - \frac{1}{(1 + (30 \div 10))}$	0.75
Number of Shipping Orders	$25 \in$ 21-30	0.75
Credibility of the Local Rating	$(0.84 - 0.54) \leq 1 \leq (0.84 + 0.54)$ $0.3 \leq 1 \leq 1.38$	1

Table 4.2.3: Local Reputation Calculation

For Transaction Factor we used same corresponding factor value in table 4.2.1. Next, based on the computed factors, we calculate the local reputation score e_t using the equation 7.

$$e_t = \frac{(1 \times 1) + 0.75 + 0.525 + 0.75}{4} = 0.75625$$

Finally, as the local reputation score for the shipping request is greater than the positive rating threshold θ, $0.75625 \geq 0.5$, we use the increase model to compute the global reputation $GRep_t$.

$$\Delta GRep_t = 0.1 \times (1 - \frac{0.88}{1}) = 0.012$$
$$GRep_t = 0.88 + 0.75625 \times 0.012$$
$$GRep_t = 0.889$$

CHAPTER 5

Design and Implementation of a Blockchain-based Crowdshipping Application

This chapter demonstrates the design and implementation of a Blockchain Crowdshipping platform that offers decentralization, rights to data accessibility, visibility, and transparency using blockchain features. The key objective is to provide a free-mediator platform that ships goods and valuables among parties who do not trust one another depending on a reputation model. Thus, increasing the effectiveness of present crowdshipping procedures by establishing traceable and immutable chains of transactions and blocks.

5.1 Operational Scenario

Parcel shipping processes include different actors with each assigned task and role. The blockchain network has an organization for each actor. The followings are the key players in the proposed system:

1. **Seller** initiates the shipping process by creating a parcel and shipping order. Seller is responsible for handing over the parcel to the courier, verifying successful parcel delivery, and providing ratings to the courier.

2. **Customer** is the entity that will receive the parcel. Customer provides the

shipping destination and delivery time window and agrees on parcel details. Also, confirms parcel receipt and give ratings to the courier.

3. **Courier** submits bids toward the orders that match his reputation and availability. Courier is responsible for parcel shipping to the correct customer.

As illustrated in Figure 5.1.1, the shipping process scenario starts with creating the parcel. Sellers can use the smart contract to create a package they want to ship to a customer.

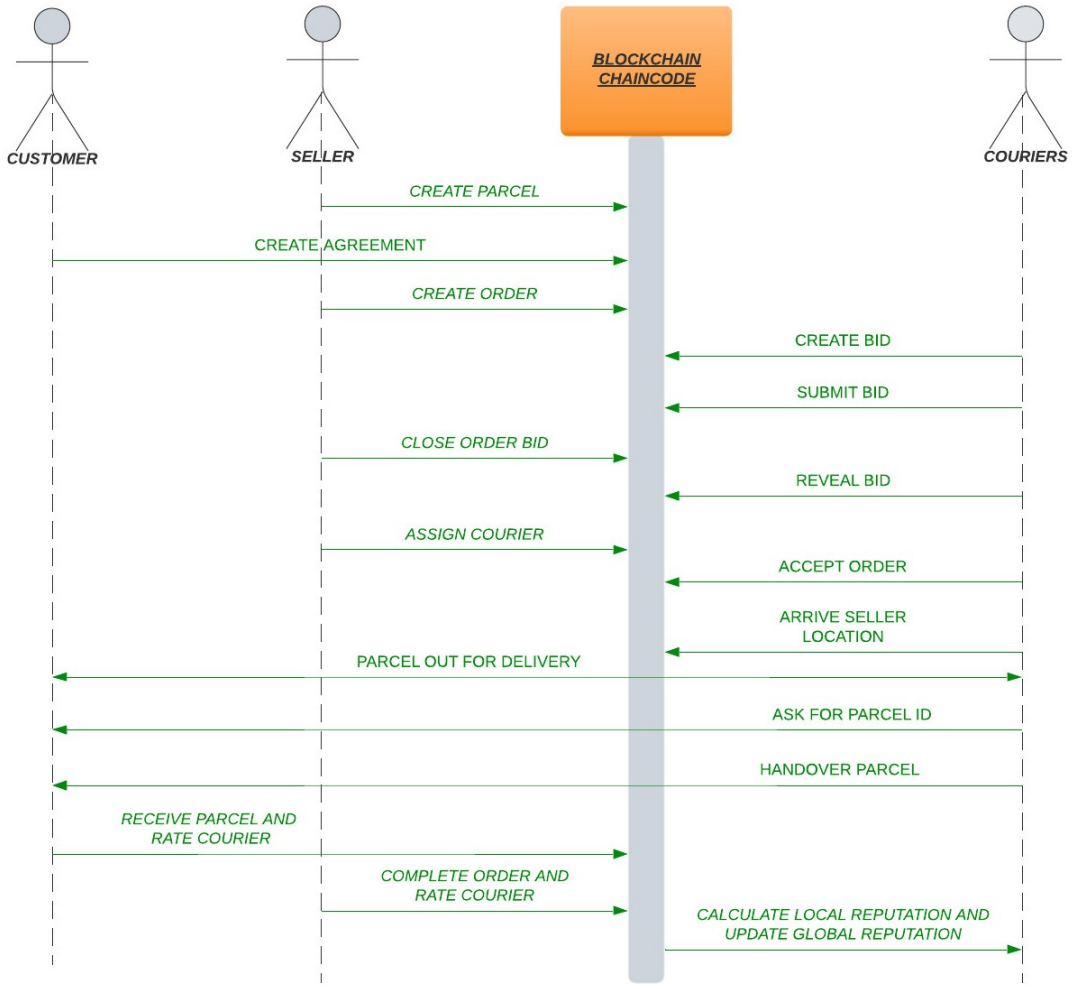

Fig. 5.1.1: Sequence diagram of successful shipping process scenario

The parcel's secret details, such as parcel size, quantity, appearance, price, etc., are stored in a private data collection and hashed to generate the parcel ID. These details

can only be accessed by the seller's organization and the seller who owns the parcel ID. Using the same function, the seller creates shared parcel data with the customer. Shared Data is stored in private data collection called Parcel Collection, accessible by the seller and customer organizations by the member who knows the parcel ID. The seller passes out the band parcel ID and the parcel properties through email or other communication to the customer. Next, the customer adds the convenient shipping destination and time and verifies the mutual package properties before agreeing to transact. The provided parcel's private properties must generate a hash that identically matches the hashed parcel ID; otherwise, this step is failed. This step ensures that the agreed parcel information is consistent with the customer's expectations.

Upon customer agreement, the seller creates an order to assign a courier to fulfill the shipping request. Any order consists of public data visible to any channel member, no matters his role and organization. This information helps the courier to evaluate his availability to do the shipping task and estimate the compensation amount in the bid. The order's public data includes the shipping date, estimated pickup and drop-off locations, minimum reputation value of the courier, and maximum amount paid by the seller. On the other hand, the private data includes the assigned courier, shipping cost, parcel ID, and other metadata. Order's confidential data is stored in private collection shared between the selected courier and the seller and protected by a complex key of the Order ID and the Transaction ID.

Each order is created with the status open. While the order is open, couriers can send their bids toward the order. Once the order is published in the world state (database), the courier can query the ledger for the open orders. If the courier is interested in a particular order, first, he creates his full bid in his organization's implicit private data collection. The full bid includes the courier identity and requested price. However, before adding the bid to his organization's implicit private data collection, the smart contract verifies that the courier's global reputation matches the minimum threshold specified in the order and the provided price is less than the maximum paid amount in the order. If the bid is created successfully, the courier submits the bid's hash to the order without revealing the requested price.

Meanwhile, the courier's organization is added to the list of organizations necessary to endorse any order's updates. After several bids have joined the order, the seller wants to close it and allow couriers to reveal their offers. The seller changes the order's state to closed to prevent additional bids from being submitted. Then, couriers can show their full bids and try to win the order.

Later, the seller assigns one of the bidders by calculating the lowest price from the set of revealed bids. Courier organization also calculates the lowest price requested by couriers. The seller can assign a courier successfully if his organization and the courier organization endorse the same courier and price. Before approving the transaction that assigns the courier, the courier organization queries the implicit private data collection on its peers to check if any courier has a lower price bid that has not been revealed yet. If found, the organization will withhold its endorsement and prevent the courier from being assigned. That prevents the seller from prematurely assigning a courier to the order or colluding with couriers with unfair prices. After Courier's assignment, the selected courier can accept or reject the order. In case he rejects the order, a certain amount of his global reputation will be deducted specified by the network.

In the next stage, the seller updates the parcel and order state to Courier Assigned and transmits the order transaction ID, the exact pickup and drop-off locations out of the chain by email, or any communication method to the courier. The courier arrives at the pickup location and notifies the seller on the chain by changing the order state. The courier picks up the parcel, and the seller updates the parcel and order state to Out For Delivery. When the courier reaches the customer's destination, he will ask him to provide the parcel ID, which is confidential between the seller and customer; if the parcel ID is correct, then the courier will successfully update the parcel to Handedover to the customer. At the same time, the customer will update the parcel state to Received by Customer and provide an evaluation score for the courier. Finally, the seller can verify the correctness of the parcel and order states, provide an evaluation score of the courier service, and set the parcel as Delivered and the order as Completed. Figure 5.1.2 summarize how the parcel, public order, and

private order status change based on the smart contract execution.

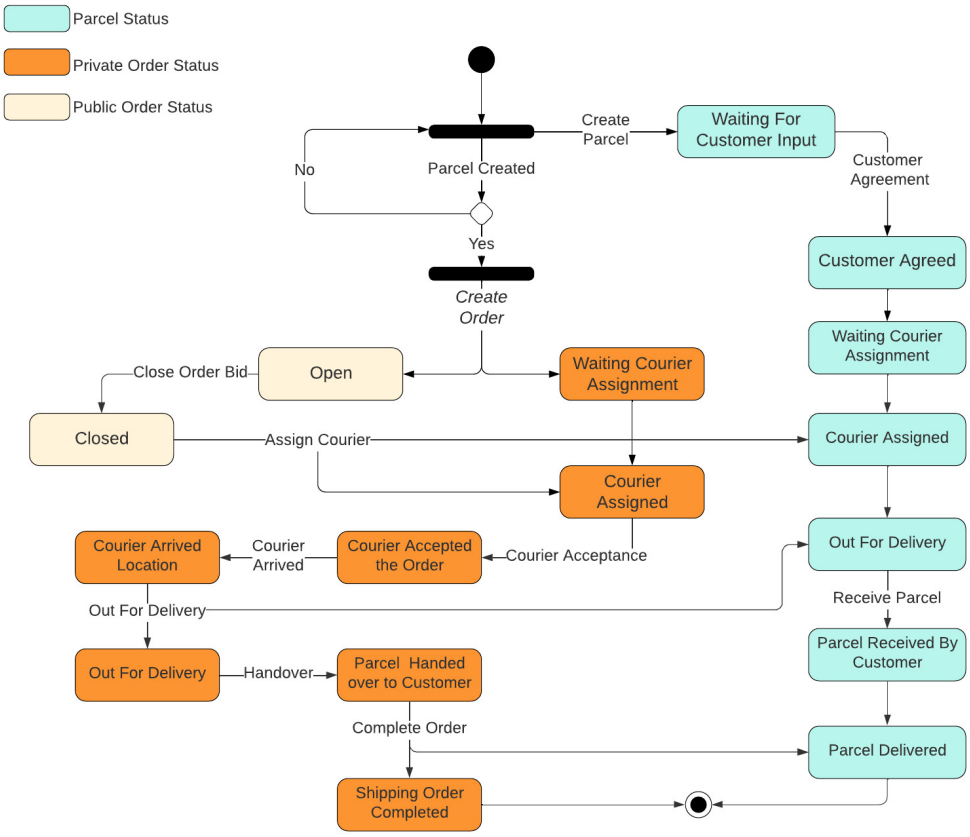

Fig. 5.1.2: Proposed System Components State Diagram

The smart contract automatically collects all required data to estimate the courier's local reputation score, including shipping cost, previous and current order timestamp, the courier's total number of transactions, and the credibility of the provided evaluation. Finally, update the courier's global reputation score based on the decrease and increase models discussed in chapter 4.

The seller has the right to call the cancellation function to cancel the transaction before the parcel is shipped. If the shipping order state change to courier arrived, that means that the shipping process has started, and cancellation is not applicable after this point.

5.2 Network Model

The blockchain network consists of four organizations, with one peer for each as depicted in figure 5.2.1. A dedicated certificate authority is assigned for each organization. The Peer node is intended to be an endorsing peer where the system's chaincode resides. Each peer maintains a current state database as the couch DB. The sequence of our proposed work is creating a channel; each peer must join the channel, install the chaincode, and approve it. If the peer receives sufficient approvals from the organization, it commits the chaincode, invokes it, queries it, and enables client communication with Postman API.

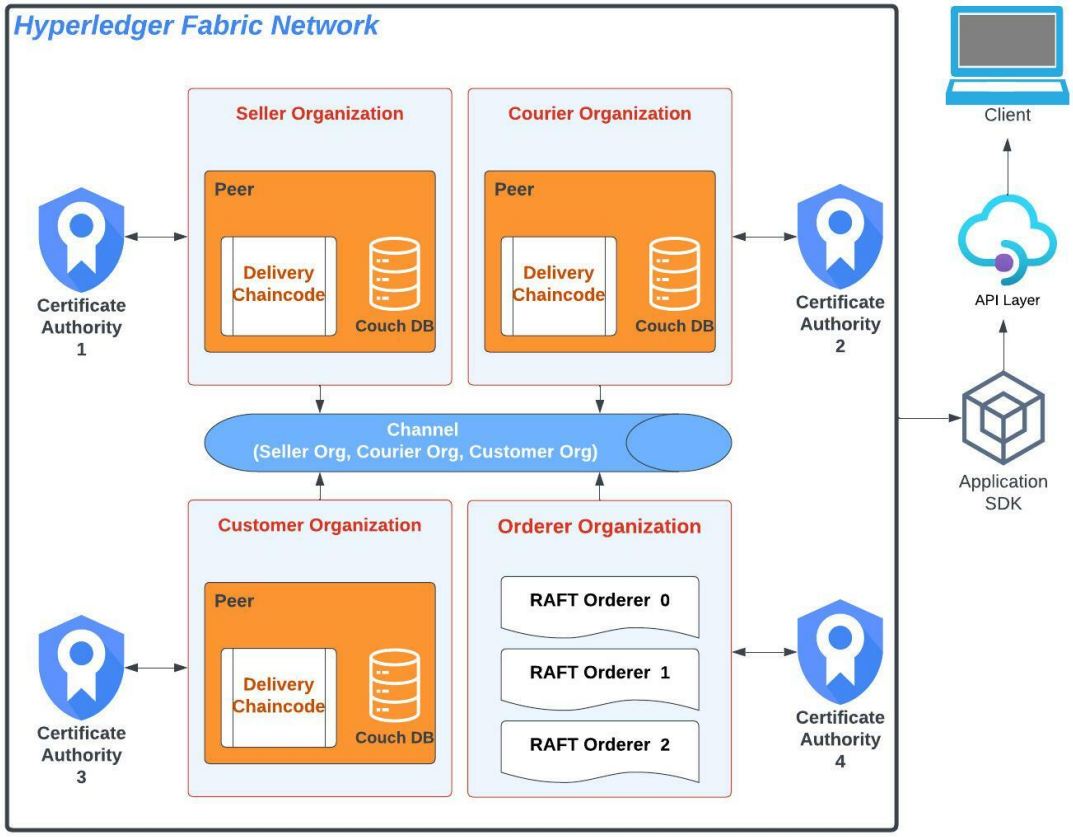

Fig. 5.2.1: Proposed LMD Platform Network Architecture

Hyperledger Fabric offers several implementations for achieving consensus between ordering service nodes, Raft, Kafka, and Solo. In our network, we use Raft over others for the following reasons. A Raft is a crash fault-tolerant (CFT) ordering service

based on an implementation of the Raft algorithm in etcd. Raft uses the "leader and follower" architecture, each channel elects a leader node, and the followers replicate that node's decisions. Due to its endorsement policy of majority vote, RAFT provides a means for high availability for ordering services. In contrast, Kafka is similar to the Raft protocol but utilizes Zookeeper ensembles and a broker, creating overhead for transaction latency and throughput. The Solo ordering service, which has one ordering node, is exclusively meant for testing, not production.

The platform's business logic comprises Chaincode, application SDK, and Program Programming Interface (API) testing tool, which work together to deliver the features of the application. Chaincode consists of the Data model designed to define the data structures necessary for the application network (Chaincode and Application SDK). These data structures are Parcel, Shipping Order, Shipping Order Private Details, Full Bid, Bid Hash, and Global Reputation. Second, the main smart contract called Order, which is written in Golang and verifies invoker roles and executes the associated transaction functions for each service capability's logic. Order smart contract consists of several functions, Create Parcel, Customer Agreement, Create Order, Bid, Submit Bid, Close OrderBid, Reveal Bid, Assign Courier, Courier Acceptance, Courier Arrived, Out For Delivery, Recieve Parcel, Parcel Handover, and Complete Order. Order Queries includes the functions that query the ledger, and Utils has predefined administrative capabilities used to verify user identity, retrieve collections, and manage endorsement policy.

The application SDK provides access to chaincode running within that blockchain network and to which transactions can be submitted or queries can be evaluated. It is written in javascript and uses the Gateway class as the entry point to the Hyperledger Fabric blockchain network. Once instantiated, this long-living object delivers a reusable connection to a peer within the blockchain network and allows access to any of the blockchain Networks channels for which that peer is a member.

For testing Program Programming Interface (API), we use the Postman tool that enables the client to interact efficiently with the system and determine how system resources are defined and addressed. It is an HTTP client that tests HTTP requests,

utilizing a graphical user interface using different endpoint interaction methods, such as:

- **HTTP GET**

  Retrieve a resource from the Blockchain world state in the form of a JSON payload.

- **HTTP POST**

  Establishes a resource by asking the application to run the required smart contract, after which the transaction's JSON payload (the outcome of the execution) is sent to the Hyperledger Fabric peer (running consensus) as a means of invoking it.

5.3 Smart Contract Implementation

This section explains the algorithmic details for the primary last-mile delivery operations like creating the parcel and order, bidding, courier assignment, and reputation building in the proposed system. Each function in the Order smart contract requires Transaction context, created when a smart contract is deployed to a channel and made available to every subsequent transaction invocation. It enables smart contract developers to access a wide range of Fabric APIs so they may carry out actions related to transaction processing. These include retrieving the digital identity of the client that submitted the transaction and querying or updating the ledger on both the mutable world state and the immutable blockchain.

5.3.1 Create Parcel

The create parcel function is restricted for the clients who register the system as Sellers, as illustrated in algorithm 5.3.1. The seller submits a proposal request to invoke create parcel chaincode function (reading or writing data) to endorsing peers who are part of the collection's authorized organization. The parcel's private data is sent in a transient field to protect the confidentiality of the data as the transaction is proposed,

endorsed, and committed to the ledger. The hashed parcel ID, parcel's state, seller's organization, and seller's identity are part of the parcel information stored on Parcel data collection shared between the Seller and Customer organizations.

Algorithm 5.3.1 Algorithm 1 Create Parcel

Input: parcel's description, size, quantity, value, clientOrgID, clientID
  if $clientID.role$ is Seller **then**
    $Parcel.ParcelID \leftarrow hash(parcel'sdescription, size, quantity, value)$
    $Parcel.SellerOrg \leftarrow clientOrgID$
    $Parcel.Seller \leftarrow clientID$
    $Parcel.ParcelState \leftarrow WaitingForCustomerInput$
    return $ParcelID$
  else
    return Error: Client is not authorized to create a parcel
  end if

5.3.2 Customer Agreement

Algorithm 5.3.2 shows the Customer Agreement function, which is restricted for the clients who register the system as Customers. The chaincode verifies that the parcel's current state is waiting for customer input . Then compares the hashed parcel ID with the generated hash of the parcel's private data agreed upon between the customer and the seller. Also, it compares the generated hash with the hash of the stored private data in the seller's implicit data collection. If any checks fail, the customer agreement will be unsuccessful. Once successful, the customer's identity, ship date, and destination are stored in the Parcel data collection. Besides, update the parcel state to Customer Agreed.

5.3.3 Create Shipping Order

When the customer adds his agreement, only the seller can create the shipping order using the Create Order function. The function verifies that the client has a Seller role and he is the parcel's seller. Besides, confirms the parcel's state is Customer Agreed. Each shipping order has private data stored on Order data collection accessible by Seller and Courier organizations and public information on the ledger. Both data are

Algorithm 5.3.2 Algorithm 2 Customer Agreement

Input: parcelID, parcel's description, size, quantity, value, ShipDate, Destination
  if *clientID.role* is Customer **then**
    if *ParcelState* == *WaitingForCustomerInput* **then**
      if *hash(parcel's description, size, quantity, value)* == *parcelID* **then**
        Parcel.ShipDate ← *ShipDate*
        Parcel.Destination ← *Destination*
        Parcel.Customer ← *clientID*
        ParcelState ← *CustomerAgreed*
      else
        return Error: Hash mismatch parcelID
      end if
    else
      return Error: Parcel State must be Waiting For Customer Input
    end if
  else
    return Error: Client is not authorized to perfrom agreement
  end if

described in algorithm 5.3.3. The order will be viewable in an Open state, allowing the couriers to know that the order accepts bids. It is worth noting that the order pickup location and destination are estimated locations (such as city and neighborhood) to maintain the privacy of the seller and customer and allow the courier to decide whether the order is feasible with the specified maximum paid amount in the order. The order's private data is stored using a complex key of orderID and OrderTxID, which are automatically generated from the function.

5.3.4 Create Bid

While the order is open, couriers can add new bids to the order using the Bid function. The function ensures that the client has the role of courier. The Bid function sends the bid information that includes offered price and courier identity in a transient field of the proposal to keep price confidentiality. While the order is open, couriers can add new bids to the order using the Bid function. The function ensures that the client has the role of courier. The Bid function sends the bid information that includes offered price and courier identity in a transient field of the proposal to keep price confidential-

ity. The function performs two checks, the courier's price is less than the maximum paid amount specified in the order, and the courier's global reputation matches the minimum reputation threshold. Each courier's full bids are stored in their organization's implicit private data collections using bidTxID, automatically generated by the function. Each courier's full bids are stored in their organization's implicit private data collections using BidTxID, automatically generated by the function. Algorithm 5.3.4 describe the Bid function.

Algorithm 5.3.3 Algorithm 3 Create Shipping Order

Input: parcelID, MinimumReputation, MaximumPaidPrice, PickupLocation
  if $clientID.role$ is Seller **then**
    if $clientID == Parcel.Seller$ **then**
      if $ParcelState == CustomerAgreed$ **then**
        $PrivateOrder.OrderID \leftarrow OrderID$
        $PrivateOrder.Seller \leftarrow clientID$
        $PrivateOrder.OrderDate \leftarrow TransactionTimestamp$
        $PrivateOrder.ParcelID \leftarrow parcelID$
        $PrivateOrder.State \leftarrow WaitingCourierAssignment$
        $Order.MinRep \leftarrow MinimumReputation$
        $Order.MaxPaid \leftarrow MaximumPaidPrice$
        $Order.PickupLocation \leftarrow PickupLocation$
        $Order.PickupDate \leftarrow Parcel.ShipDate$
        $Order.ShippingLocation \leftarrow Parcel.Destination$
        $Order.PrivateBids \leftarrow null$
        $Order.RevealedBids \leftarrow null$
        $Order.State \leftarrow Open$
        return OrderID, OrderTxID
      else
        return Error: Parcel State must be Cutsomer Agreed
      end if
    else
      return Error: Client is not the parcel owner
    end if
  else
    return Error: Client is not authorized to create order
  end if

Algorithm 5.3.4 Algorithm 4 Create Bid

Input: OrderID, price, clientID, clientOrg
  if $clientID.role$ is Courier **then**
    if $Order.State == Open$ **then**
      if $Price \leq Order.MaxPaid$ **then**
        if $clientID.GlobalReputation \geq Order.MinRep$ **then**
          $Bid.Courier \leftarrow clientID$
          $Bid.Org \leftarrow clientOrg$
          $Bid.Price \leftarrow price$
          return BidTxID
        else
          return Error: Courier's global reputation is less than order's minimum reputation
        end if
      else
        return Error: Offered price is greater than seller's maximum paid amount
      end if
    else
      return Error: Failed to create Bid, order must be Open
    end if
  else
    return Error: Client is not authorized to create a bid
  end if

5.3.5 Submit Bid

After the bid's creation, the courier can submit the hash of the bid toward the order using Submit Bid function, demonstrated in algorithm 5.3.5. First, the function assures that the client has the role of Courier and the order state remains Open. Next, add the courier bid to the order's private bids, and the courier's organization is added to the list of organizations that need to endorse any updates to the order.

5.3.6 Close Bid

The seller closes the order to prevent additional bids from being added to it. The function validates that client is the order's seller and order's state is Open. Only Order State is updated to Closed.

Algorithm 5.3.5 Algorithm 5 Submit Bid

Input: OrderID, BidTxID
  if $clientID.role$ is Courier **then**
    if $Order.State == Open$ **then**
      $Hash \leftarrow BidHash(BidTxID)$
      $Order.PrivateBids \leftarrow Hash$
      $Order.Orgs \leftarrow clientOrg$
    else
      return Error: Failed to submit Bid, order must be Open
    end if
  else
    return Error: Client is not authorized to submit a bid
  end if

5.3.7 Reveal Bid

After closing the order, couriers now use the Reveal Bid function to reveal their bids. Algorithm 5.3.6 shows that the function validates that the transaction submitter is a courier, then it needs to satisfy the following conditions:

1. The order is closed.
2. The transaction was submitted by the identity of the courier who created the bid.
3. The revealed bid's hash matches the bid's hash on the channel ledger to confirm that the bid is the same as the bid stored in the private data collection.
4. The hash of the revealed bid matches the hash submitted to the order to ensure that the bid has not altered after closing the order.

5.3.8 Assign Courier

The seller uses this function to assign a courier out of the candidates. The function calculates the lowest price offered from the set of revealed bids. If two or more bids had the same price, the first one who submitted the bid would be assigned. Courier organization also calculates the price that clears the order and the best courier. The seller can assign a courier only if the Courier organization endorses the same courier

and price. It queries the implicit private data collection on their peers to check if any courier has a bid that has yet to be revealed and has a lower price. If found, the organization will withhold its endorsement and prevent a courier from being assigned. This approach prevents the seller from assigning the order's courier prematurely or colluding with a courier at an artificially low price. Algorithm 5.3.7 represents the Assign Courier function.

Algorithm 5.3.6 Algorithm 6 Reveal Bid

Input: OrderID, BidTxID, price, clientID, clientOrg
  if $clientID.role$ is Courier **then**
    if $Order.State == Closed$ **then**
      if $Bid.Courier == clientID$ **then**
        if $Order.PrivateBids == hash(price, clientID, clientOrg)$ **then**
          if $Order.PrivateBids == BidHash(BidTxID)$ **then**
            $Order.RevealedBids.Price \leftarrow price$
            $Order.RevealedBids.Courier \leftarrow clientID$
            $Order.RevealedBids.Org \leftarrow clientOrg$
          else
            return Error: Failed to reveal bid, order bid mismatch the hash of bid private data
          end if
        else
          return Error: Failed to reveal bid, bid hash mismatch the generated hash of transient data
        end if
      else
        return Error: Client is not the courier who owns the bid
      end if
    else
      return Error: Failed to reveal bid, order must be closed
    end if
  else
    return Error: Client is not authorized to reveal a bid
  end if

Algorithm 5.3.7 Algorithm 7 Assign Courier

Input: OrderID, OrderTxID, ParcelID
  if $clientID.role$ is Seller **then**
    if $Order.State == Closed$ **then**
      $n \leftarrow count(Order.RevealedBids)$
      $LowestPrice \leftarrow 0$
      $AssignedCourier \leftarrow null$
      for $i = 0$ to n **do**
        if $RevealedBids[i].Price > RevealedBids[i+1].Price$ **then**
          $LowestPrice = RevealedBids[i+1].Price$
          $AssignedCourier = RevealedBids[i+1].Courier$
        else
          $LowestPrice = RevealedBids[i].Price$
          $AssignedCourier = RevealedBids[i].Courier$
        end if
      end for
      $PrivateOrder.ShippingCost \leftarrow LowestPrice$
      $PrivateOrder.Courier \leftarrow Courier$
      $PrivateOrder.State \leftarrow CourierAssigned$
      $Parcel.State \leftarrow CourierAssigned$
    else
      return Error: Failed to assign courier, order must be closed
    end if
  else
    return Error: Client is not authorized to assign courier
  end if

5.3.9 Courier Acceptance

The assigned Courier will be notified that he has been selected to fulfill the order. Now, he needs to provide his acceptance to perform the shipping task. If he rejects the order, the courier loses a predefined amount of his global reputation specified by the network administrator. Besides, the function automatically assigns another courier to the order, similar to the Assign Courier function.

5.3.10 Courier Arrived

The courier confirms his arrival at the parcel's pickup location. The function is accessible to users with a courier role and compares the submitting client with the

assigned courier in the private order. If both match, the function updates the order's state to Courier Arrived.

5.3.11 Out For Delivery

After the courier's arrival, the seller hand the parcel to the courier and updates the order and the parcel state to Out For Delivery. This function ascertains that the invoker is the order's seller and the current order's state is Courier Arrived. The function updates the order's state to Out For Delivery if both conditions are met.

5.3.12 Handover

When the courier arrives at the customer's destination, he uses this function for two purposes. First, it proves that the receiver is the actual parcel's customer. As the courier will ask the customer to provide him with the parcel ID, if it matches the parcel ID in the order's private details, he will hand it to the correct person. Next, the function automatically updates the order's state to Parcel Handedover to Customer.

5.3.13 Receive Parcel

The customer uses this function to prove the parcel receipt after verifying the client's identity. To update the parcel's state to Parcel Received by Customer, the parcel's state must be Out For Delivery; otherwise, the customer will be denied from achieving this step. The customer can also leave a rating to the courier at this step.

5.3.14 Complete Order

The final step taken in the shipping process is completing the order . The seller update the orders' state to Order Completed and parcel's state to Parcel Delivered. At the beginning, the function qurey the order's private details to confirm that submitting client is the order's seller and the state is Parcel Hanedover to Customer.

The Seller evaluate the courier performance by providing a rating, which could be 1 or -1. The function automatically invoke a function called Calculate Reputation. It gather all factors value discussed in chapter 4 and the outcome is the local reputaion. Once obtained, it modifies the courier's global reputaion score, append the calcualted local reputation and increases the courier's total number of shipping requests, update rating's average, standard deviation, and Previous order timestamp. Algorithm 5.3.8 depicts Complete Order function.

Algorithm 5.3.8 Algorithm 8 Complete Order

Input: OrderID, OrderTxID, ParcelID, RatingScore
  if *clientID.role* is Seller **then**
    if *clientID* == *PrivateOrder.Seller* **then**
      if *PrivateOrder.State* == *ParcelHandedovertoCustomer* **then**
        Function *CalculateReputation* {Courier, PreOrderDate, OrderDate, ShippingCost, RatingScore}
        LocalRating ← Calculate Local Reputation Score **see section 4.2.2.1 for reputation score calculation**
        GlobalRating ←Calculate Global reputation Score **see section 4.2.2.2 for reputation score calculation**
        return LocalRating
        End Function
      else
        return Error: Failed to complete order, order's state must be Parcel Handedover to Customer
      end if
    else
      return Error: Client is not the order's seller
    end if
  else
    return Error: Client is not authorized to complete the order
  end if

5.3.15 Cancel Order

This function enables the seller to cancel the order at any time before the courier starts the shipping task. First, it validates that the client has a seller role and he is the order's seller. Then, it checks if the order's state is Open or Closed and Private Order's state is one of these states, Waiting Courier Assignment, Courier Assigned, or

Courier Accepted the Order. Otherwise, canceling the order is inapplicable. Finally, the function update both Order and Private Order state to Cancelled.

Queries functions and utils used to invoke the smart contracts successfully are available on Github [4] .

This chapter described the design and implementation of the Blockchain LMD application's capabilities. Furthermore, this chapter reveals how the system architecture components are relational dependencies. Furthermore, the access control and conditions were defined and coded in the Smart Contract code to meet the shipping requirements with whom and what each entity can do in the application to ensure the correctness of the smart contracts execution work.

CHAPTER 6

Discussion and Results

This chapter presents a feature-based comparison between our work and related work reviewed in 3. Security, privacy, and scalability aspects are discussed as well. Then, we show reputation test scenarios to evaluate the effectiveness of our reputation model, then assess the application performance using Hyperledger Caliper, and discuss the results obtained.

6.1 Security Analysis

The suggested LMD solution leverages key security features from blockchain by design. Such as decentralized trust, integrity, non-repudiation, and availability. Although existing blockchain and smart contract technologies still have performance and security threats, we assume that the decentralized feature of the BC makes it impossible for an adversary to compromise the BC network and alter the ledgers' contents. Each actor in the system has a digital identity encapsulated in an X.509 digital certificate issued by a Certificate Authority (CA). These identities determine the permissions over resources and access to information. Also, by design, our framework is secured against Man-In-The-Middle (MITM) attacks and replay attacks, as every message exchange is cryptographically signed and timestamped, which ensures that nobody can repudiate their activities later. Integrity is essential in preventing critical information tampering. The proposed framework provides the ability to use transaction logs to track back historical occurrences. To guarantee proof of delivery linked to the parcel, we generated the parcel ID as a hash of all parcel properties; any

change in the parcel characteristics will cause a difference in the hash. The parcel ID is requested from the customer to complete a successful verification. Additionally, a reputation was devised to hold couriers responsible and discourage fraud in the suggested system. Every courier needs to maintain a minimum reputation score to be eligible to receive shipping requests. Additionally, couriers must stay active and continue to fulfill shipping requests, as reputation decay if they are inoperative.
Table 6.1.1 compares the examination of several features offered in previously explored works in chapter 3 to the proposed system. We examine each system using standard security criteria and offered features:

- **Accountability**: Each participating entity must take responsibility for every delivery operation they carry out.
- **Auditability**: a delivery state is systematically and independently examined to ascertain whether the shipping operation is proper (following the consistency requirements) and has always been correct.
- **Anonymity**: participants' identity in a smart contract is assured to remain anonymous. Additionally, all shipping-related information is kept secret.
- **Courier Reputation**: The solution assign a reputation score to each courier that will be considered during the selection process
- **Proof of Delivery**: The intended recipient receives the package, and no other party may effectively assert a claim to it.
- **Traceability**: The solution must be traceable enough to identify the delivery process's critical flaw.
- **Scalability**:The scheme have a capacity to scale to meet the demands of an increasing workload while maintaining appropriate performance.

System Name	Accountability	Auditability	Anonymity	Courier Reputation	Proof of Delivery	Traceability	Scalability
Lelantos	✗	✗	✓	✗	✓	✗	✗
Single and Multiple Transporters	✓	✓	✗	✗	✓	✓	✗
DeM-CoD	✓	✓	✗	✗	✓	✓	✗
Auditable Protocols for Fair Payment	✓	✓	Only for Customers	✗	✓	✓	✗
Proposed System	✓	✓	✗	✓	✓	✓	✓

Table 6.1.1: Features Comparison Between Related Work and The Proposed Solution

The suggested trust system faces security risks from common reputation-related threats. In our system, a seller and customer must perform a complete shipping request with the courier in order to be able to provide a rating. The suggested reputation strategy itself can thwart several reputational attacks. For instance, Whitewashing is mitigated because a central authority registers participants to the permissioned network. If identity is revoked, a participant can only rejoin with the network administrator's permission. Additionally, since registration requires a form of identification, Sybil's attacks are prevented. According to prior test cases, slandering and traitor attacks are also eliminated.

6.2 Development Environment

The requirements and specification of our proposed network has been shown in Table 6.2.1. The following steps are taken in order to run the application:

1. Generate Crypto Materials for Seller Org, Customer Org, Courier Org and RAFT Orderer. It creates the node organization unit materials related to CA, MSP, peers, TLSca, admins and users for all organizations.

2. Create Channel Artifacts auch as genesis block and channel transaction files, and anchor peers. The policies are customized for reading, writing, and endorsements by the majority out of three organizations; at least two organizations must approve.

3. Creating and Joining Channel

4. Delivery Chaincode Deployment

5. Install, Approve, Commit, and Invoke Delivery Chaincode.

6. Launch the postman API server in order to allow the interaction with the application and the Hyperledger Fabric's local environment.

Requirements	Specification
Operating System	Ubuntu Linux 20.04 (8 GB RAM)(64 bit)
cURL Tool	7.68.0
Docker engine	20.10.18
Docker Composer	1.25
Go	1.14.1
Node JS	13.14.0
NPM	6.14.4
Hyperledger Fabric	2.1.1
VS Code	1.74.1
Postman API	9.31.25
Hyperledger Caliper	0.5.0
Couch DB	0.4.20
Certificate Authority	1.4.7

Table 6.2.1: Requirements and specification of proposed LMD Blockchain network

6.3 Reputation Model Testing Results

Based on the Reputation model defined in Section 4.2.2, this section will evaluate the reputation model's effectiveness in six different scenarios. Each scenario examines the significance of each factor utilized in the model, except the sixth test case, which compares our reputation algorithm to the average reputation model. Before demonstrating the scenarios and their results, it is worth noting that the parameters controlling the Reputation model must be optimized per the business requirements due to its influence on the global reputation score growth or dwindling speed.

Figure 6.3.1 shows the effect of the parameters on the global reputation score. When the increase rate is high ($\alpha = 0.1$), that will cause a faster increase in the global reputation than when α equals 0.05. For example, after 30 shipping requests,

the global reputation reached 0.9 when α was 0.1, while it scored 0.79 when α was 0.05. On the other hand, when the decrease rate is high, the global reputation will drop significantly, and a lesser decrease rate will leave a slower decline in the global reputation. We can observe that when $\beta = 3$, the global reputation bottoms out very quickly, almost within ten shipping requests. When $\beta = 1.5$, the global reputation reaches the minimum reputation after 22 shipping requests.

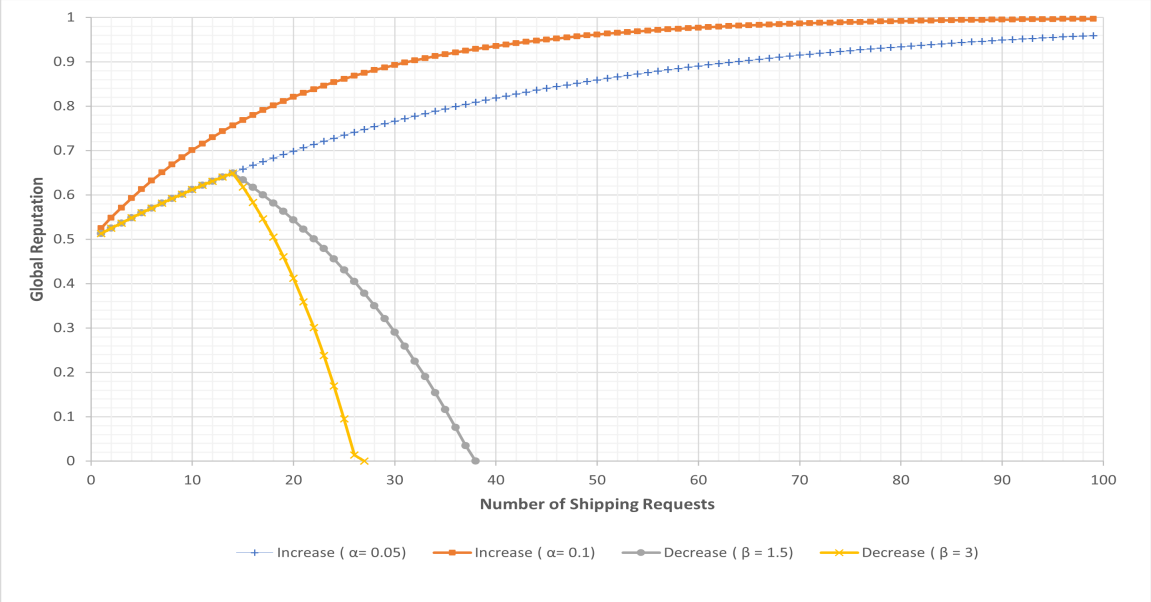

Fig. 6.3.1: Increase and Decrease Parameters Effect in Global Reputation

Our data was chosen from a unique dataset of ride-hailing journeys made between June 2, 2016, and April 13, 2017, produced by RideAustin, a nonprofit ridesharing service based in Austin, Texas [10]. The dataset consists of several features, we only select ride ID, Rider ID, Driver ID, Rating, Order Creation Time, and the Ride Cost. Table 6.3.1 shows the parameters configuration used in the following experiments to assess our proposed reputation model.

The first test case examines the reputation model resilience to the malicious behavior of the courier if he colludes with a seller and performs consecutive requests to boost his reputation score. Since the reputation algorithm considers the timestamp difference, the timestamp factor will be low if it detects consecutive requests. Figure 6.3.2 depicts two couriers with 100% positive ratings and 50 shipping requests

Parameter	Value
TX_N	1-25 requests 0.25 26-50 requests 0.50 51-75 requests 0.75 $>$ 76 requests 1
Normal time difference between two consecutive requests T_a	1 hour
Local reputation threshold θ	0.5
Maximum increase value α	0.1
Decrease rate β	1.6

Table 6.3.1: Reputation Test configuration

with shipping costs that range between 10-25$. The critical difference is that the malicious courier fulfilled requests at intervals of less than an hour. In contrast, the non-malicious courier has more than one hour difference between requests. We observe that the model can detect this behavior and control the courier's reputation. The non-malicious courier's global reputation gradually increased and reached 0.98, while the malicious remained in a range of 0.50 - 0.55.

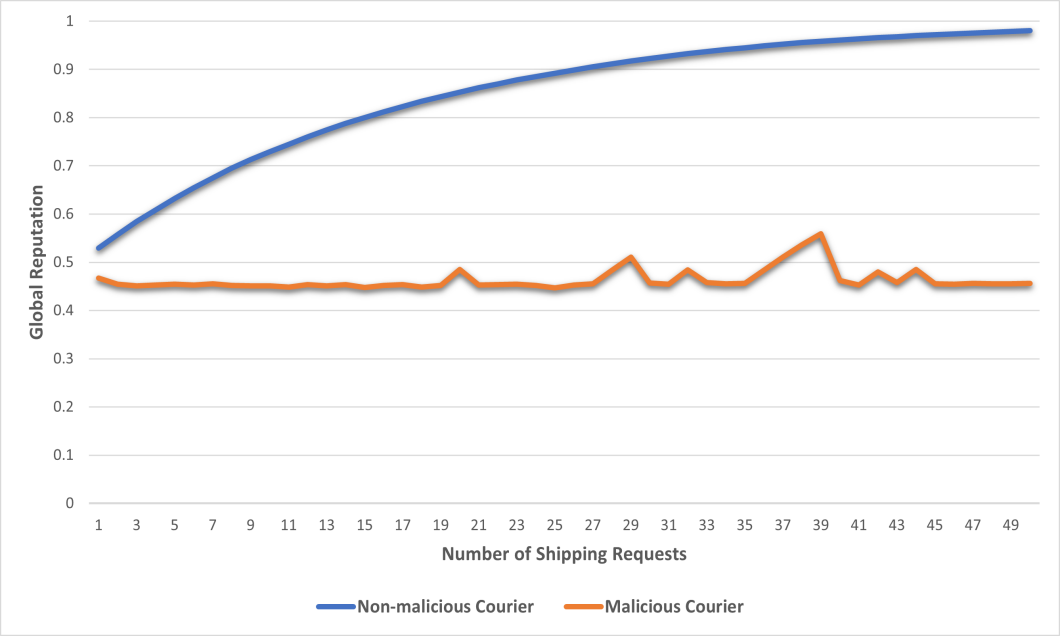

Fig. 6.3.2: Test Case 1 - Timestamp Factor Impact on Courier's Global Reputation

The second test case analyzes the reputation model action toward unfair ratings. Suppose a courier constantly receives unsatisfactory ratings from a particular user, whereas the courier provides satisfactory service to other users. In that case, the algorithm should decrease the credibility weight of the provided rating, thus reducing the impact of a negative rating on the courier's reputation. Figure 6.3.3 presents two identical couriers who accomplished 50 requests with a positive rating rate of 70%, and 30% are negative ratings. The first reputation is computed using our proposed computation and reached 0.83; the other is without the credibility factor and attained. After fifty shipping requests, the attained reputation for the first and the second courier reached 0.83 and 0.78, respectively. We notice that using the credibility factor alleviates the negative rating impacts provided unfairly by the same user.

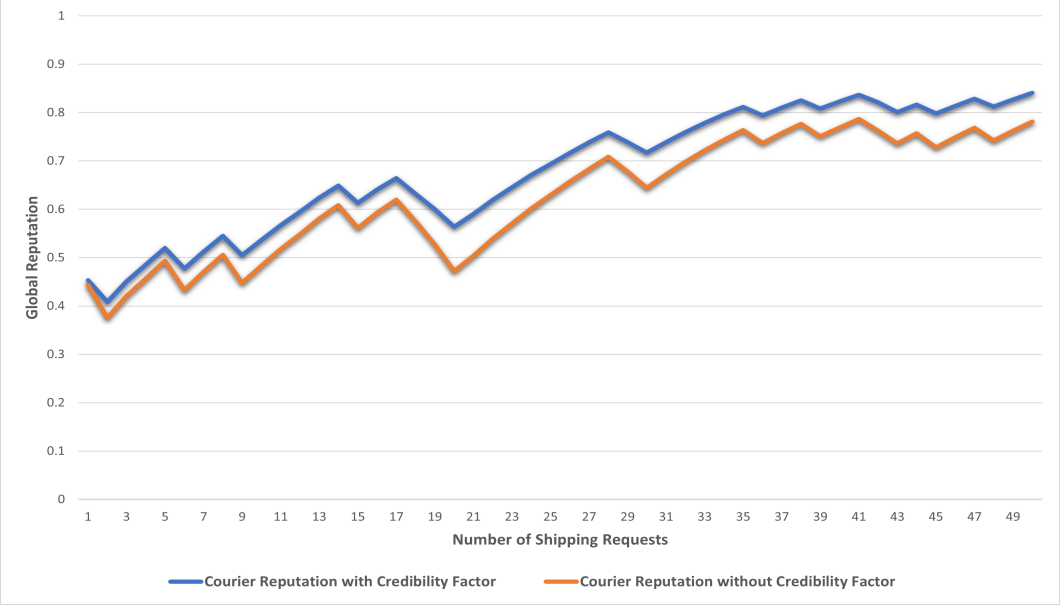

Fig. 6.3.3: Test Case 2 - Credibility Factor Impact on Courier's Global Reputation

The third test scenario concerns couriers who act dishonestly and provide unsatisfactory service or perform a traitor attack (build a robust reputation, then start to deceive system users). Figure 6.3.4 demonstrates a courier who acted honestly for the first 30 shipping requests and constructed a high reputation that reached 0.93, then began to conduct unsatisfactory requests and receive negative ratings. Both couriers

take 30 consecutive satisfactory shipping requests to increase their reputation from 0.5 to 0.93, but when the decrease rate equals 1.6, the courier takes only 15 unsatisfactory shipping requests to drop from 0.93 to 0.5. As the courier continued to act maliciously, his reputation diminished to 0.1. In contrast, when we use a decrease rate that equals the increase rate (0.5), the reputation will decline gently. In this case, it cost the courier 20 unsatisfactory requests to get his reputation declined from 0.92 to 0.87. From this perspective, our proposed algorithm shows that it is difficult to gain a reputation but easy to lose it. This will encourage the courier to always act honestly; otherwise, it would cost them a significant portion of their reputation if they perform malicious behavior.

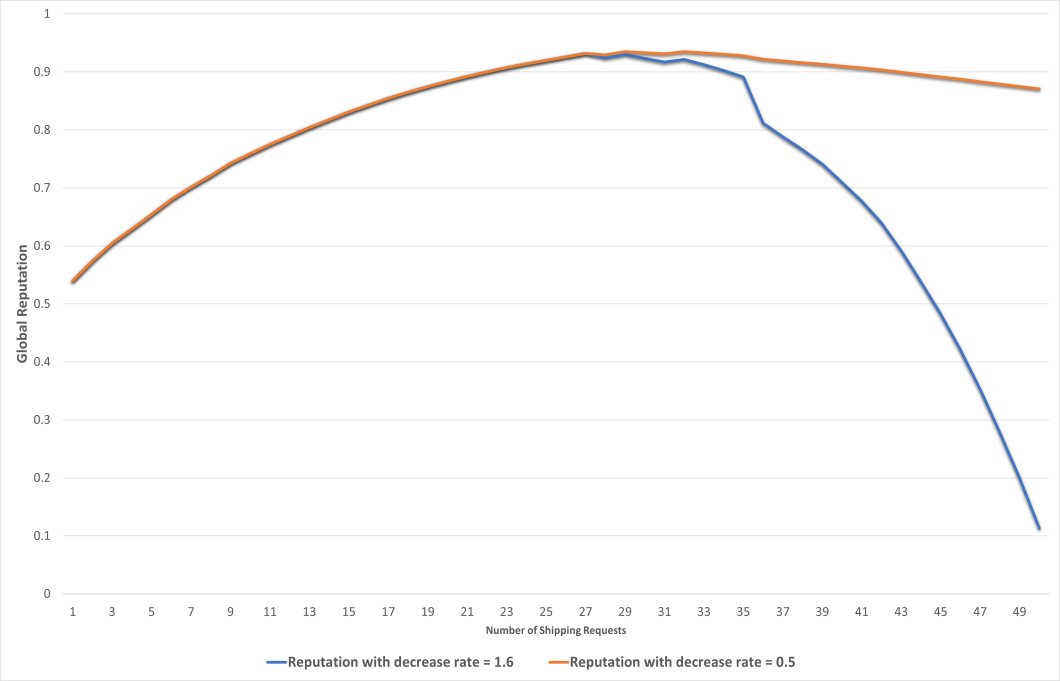

Fig. 6.3.4: Test Case 3 - Decrease rate Impact on Courier's Global Reputation

The fourth test case investigates the influence of shipping cost over reputation. If a malicious courier colludes with a seller to perform fake low-priced shipping requests, our model assigns little weight to the shipping cost factor. Figure 6.3.5 depicts two couriers with identical data except for the shipping cost. All the shipping costs of the malicious courier's requests are set to 3$, while the non-malicious courier to 10$. The cause of the steady reputation of the malicious courier for the first 25

requests is that the local reputation score is less than the specified threshold that distinguishes between satisfactory and unsatisfactory requests. When the malicious courier completed more than 25 requests, the number of transactions factor became 0.50, which made the local reputation score higher than the specified threshold. Later, the courier's reputation starts to increase gradually, similar to any honest courier, which indicates that detecting low-priced requests malicious behavior using the model is temporal. Once the courier has a higher weight of the other factors, it would be undetectable.

Fig. 6.3.5: Test Case 4 - Shipping Cost Factor Impact on Courier's Global Reputation

The fifth test shows how the reputation grows with increasing the total number of shipping requests. Intuitively, the reliable reputation model must distinguish couriers who have joined the network recently and earned a couple of good ratings from couriers who have a strong reputation built over plenty amount of shipping requests. Therefore, figure 6.3.6 presents two couriers; the first completed 50 shipping requests while the other 25 shipping requests. The first courier's reputation value is 0.98, while the other is 0.89. The weight of the number of shipping requests factor will increase as long as the courier fulfills more shipping requests. Consequently, his global reputation

will thrive. In the production setting, this factor should updated according to the highest number of shipping requests achieved in the network. Moreover, the metric could be extended to include the courier joining and how long he has been serving. Besides, the metric might be extended to include the courier network's joining and how long he has been operating in the system.

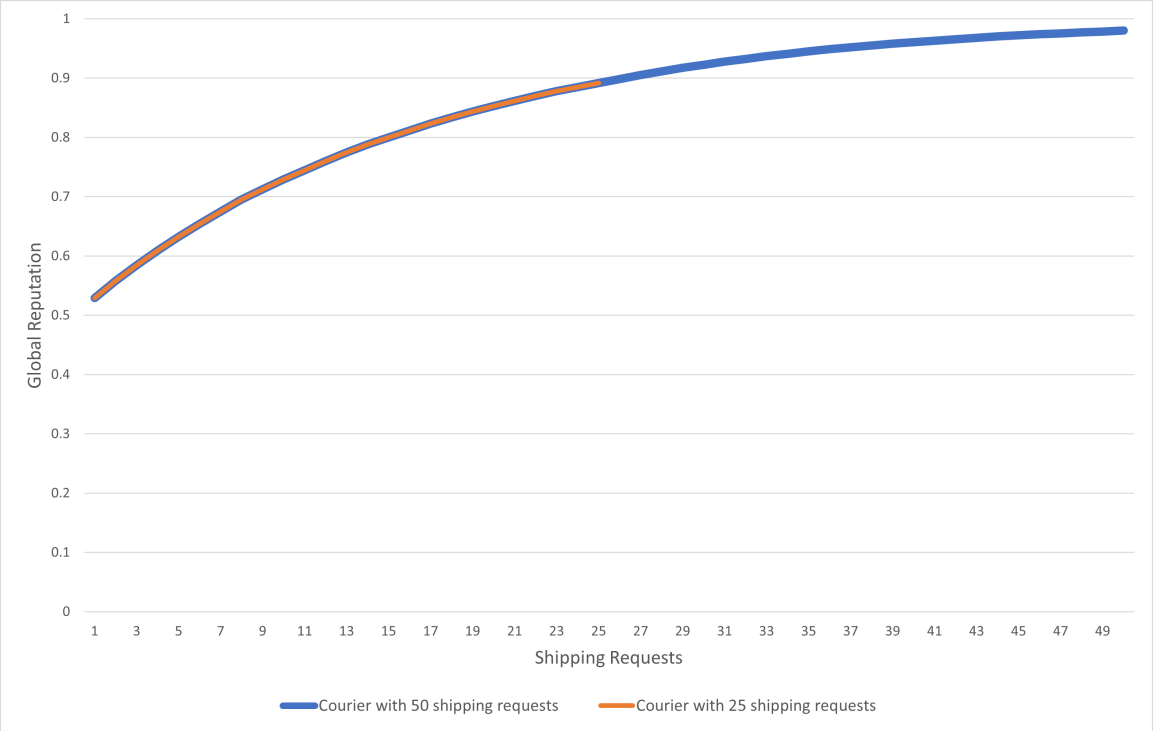

Fig. 6.3.6: Test Case 5 - Number of Shipping Requests Factor Impact on Courier's Global Reputation

The last test case compares our proposed reputation model to the Average reputation algorithm. Uber Eats [85] and DoorDash [21] are food delivery platforms that employ the Average reputation system, which depends on customer feedback. When the carrier finishes delivering the item to the customer, the customer provides a rating to express his satisfaction with the delivery service. The rating could be in a format of thumbs up (+1) and down (0) or a score ranging from 1-5 stars calculated from the 100 most recent delivery requests. The former format is selected for the comparison. In figure 6.3.7 both couriers completed 50 shipping requests with 80% positive and 20% negative ratings.

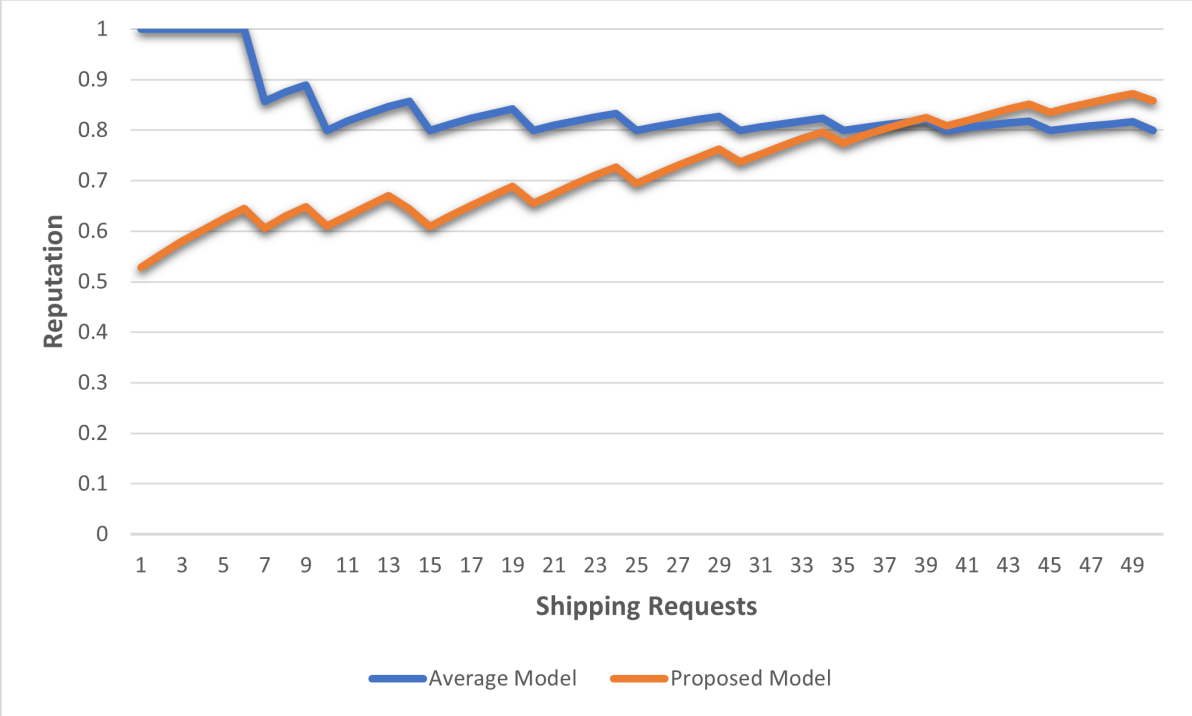

Fig. 6.3.7: Test Case 6 - Proposed Reputation Model vs Average Reputation Model

In the Average approach, we notice that if the courier always receives positive ratings, his overall reputation will remain constant at 1. This constancy of reputation causes a challenge to distinguish a new courier from a courier who has built a robust reputation by always obtaining positive ratings. Whenever a negative rating is received, both models' reputation starts to decrease. However, the impact of the negative ratings on the overall reputation of the average method falls if its portion is low compared to the positive ratings. For example, after the seventh shipping request, the courier gained a negative rating, dropping his reputation from 1 to 0.85. On the other hand, on shipping request number 40, the reputation diminished from 0.82 to 0.8.

In our proposed model, the rating is not the sole factor influencing reputation. The reduction depends on the local reputation value, as explained in the decrease model in chapter 4. Besides, the reputation gradually grows from 0.5 to 0.85, while using the Average model, it remains approximately in the same range of 0.85 - 0.8 after the first drop. Although the Average algorithm is simple and easy to use, it

is a way to summarize the data and doesn't reveal the courier's actual reputation. We believe that our proposed model has better performance. First, it is capable of differentiating between new couriers and old couriers. Second, it doesn't maintain a steady result. It is affected by the local reputation value; if it is beyond the threshold, the global reputation grows, and vice versa.

6.4 Performance Evaluation

Blockchain systems utilize a Blockchain performance measuring tool called Hyperledger Caliper (HL Caliper) [40] to evaluate the system based on several performance indicators. The send TPS rate (Transactions Per Second), transaction latency, throughput, and resource consumption are these indicators. Developers from Huawei, Oracle, IBM, and other corporations formed Hyperledger Caliper as an open source in its early stages. This study carries out the experiments locally using the Hyperledger Caliper tool, and the benchmarking indicators used by Caliper are as follows:

1. **Success Rate**: This indicator shows how many submitted transactions have been successfully processed and added to the Blockchain. A failed transaction may result from network configuration errors, time-outs, or problems in the application's chaincode.

2. **Transaction/Read Throughput**: This reveals how many transactions (or requests) your underlying blockchain network processes each second:

$$Throughput = \frac{Total\ Committed\ Transactions}{Totaltime(s)}$$

3. **Transaction/Read latency**: This statistic measures how long it takes for a transaction or query to be processed and recorded on the ledger once the client submits it:

$$Latency = Committed\ Time - Submit\ Time$$

To demonstrate the performance evaluation, the following experimental setting was configured:

1. Acquire the Caliper CLI and execute a bind command through the CLI. This step pulls the specified version of SDK packages for the selected platform (Hyperledger fabric).

2. Set up a Network configuration file to specify the LMD application access points and other necessary information, such as cryptographic materials required to interact with the Hyperledger Caliper tool.

3. Configure test files to specify test flow, workloads, and other global configuration items. For example, the type and number of clients used for the test, test rounds, rate controller, number of transactions, and arguments passed to the specified script.

Two experiments are performed using the Fixed Rate Controller. The most fundamental controller is the fixed rate controller, which is the default selection when no controller is given in the test configuration file. This controller will submit the transactions at a predetermined interval, denoted as TPS (transactions per second). for each experimental scenarios we evaluate the system performance under two different send TPS rate; at 20 TPS and 40 TPS. Five different network loads (Total Transactions) are selected to investigate the TPS rate impact on the platform's network performance in terms of throughput and latency.

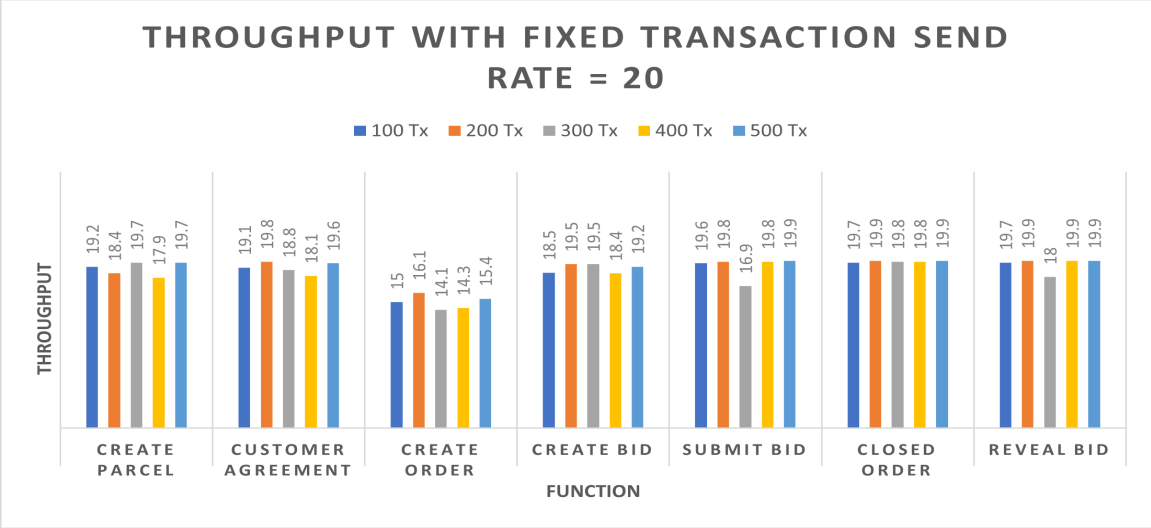

Fig. 6.4.1: Experiment I - Throughput with TPS = 20

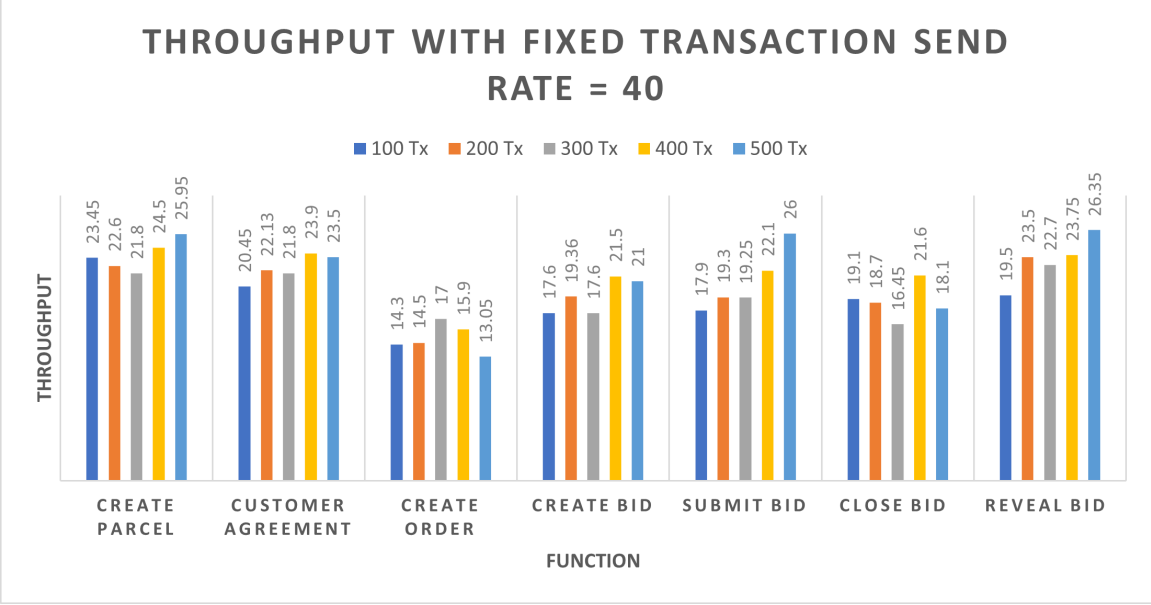

Fig. 6.4.2: Experiment II - Throughput with TPS = 40

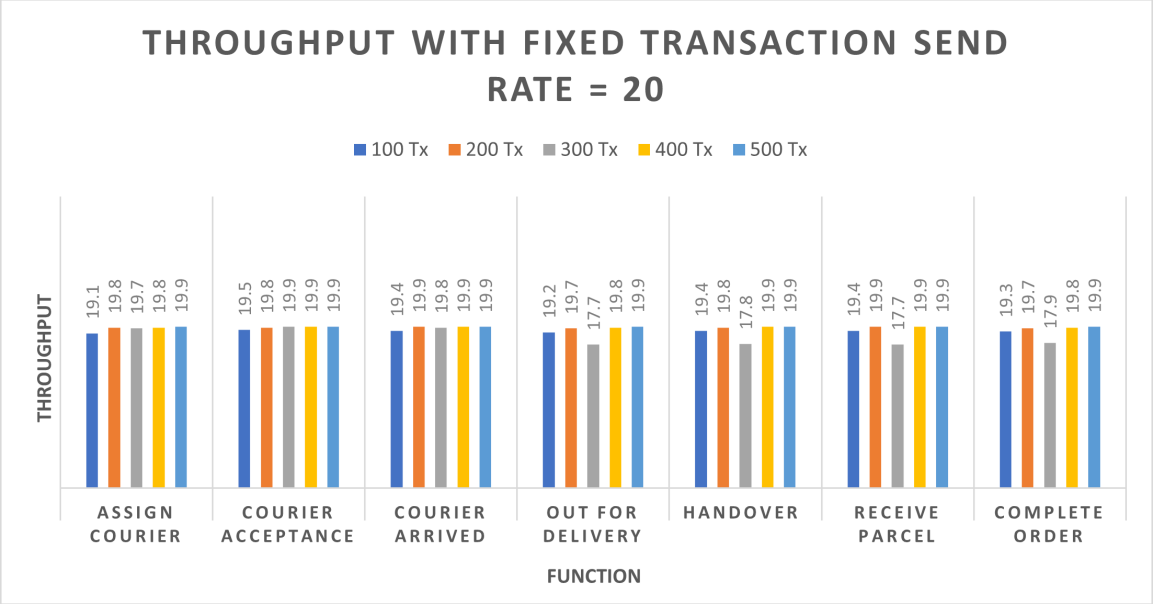

Fig. 6.4.3: Experiment I - Throughput with TPS = 20

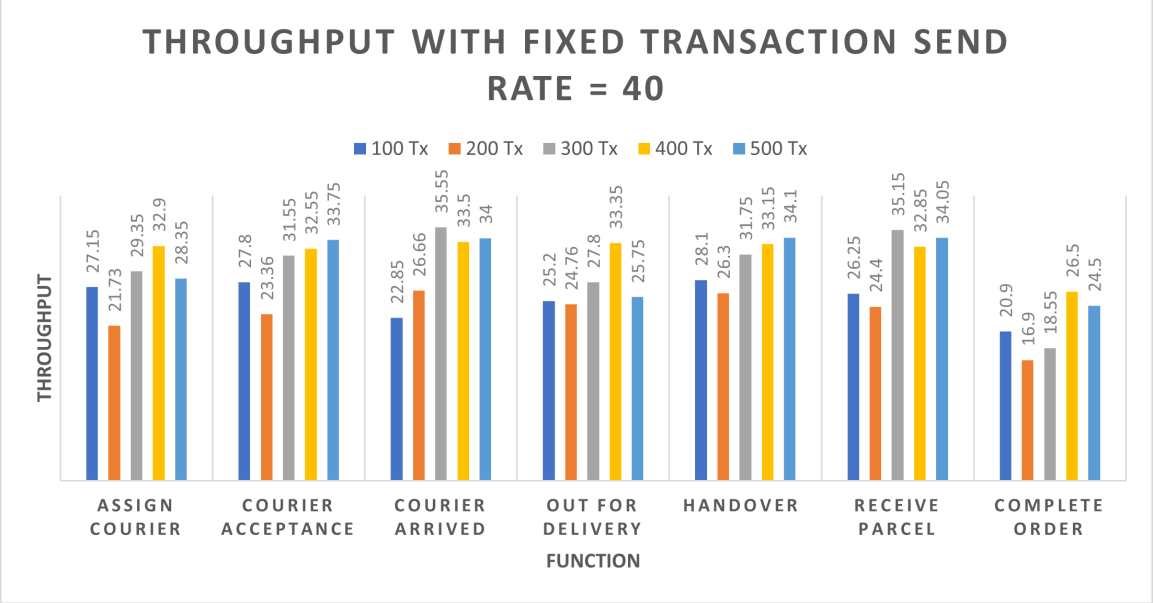

Fig. 6.4.4: Experiment II - Throughput with TPS = 40

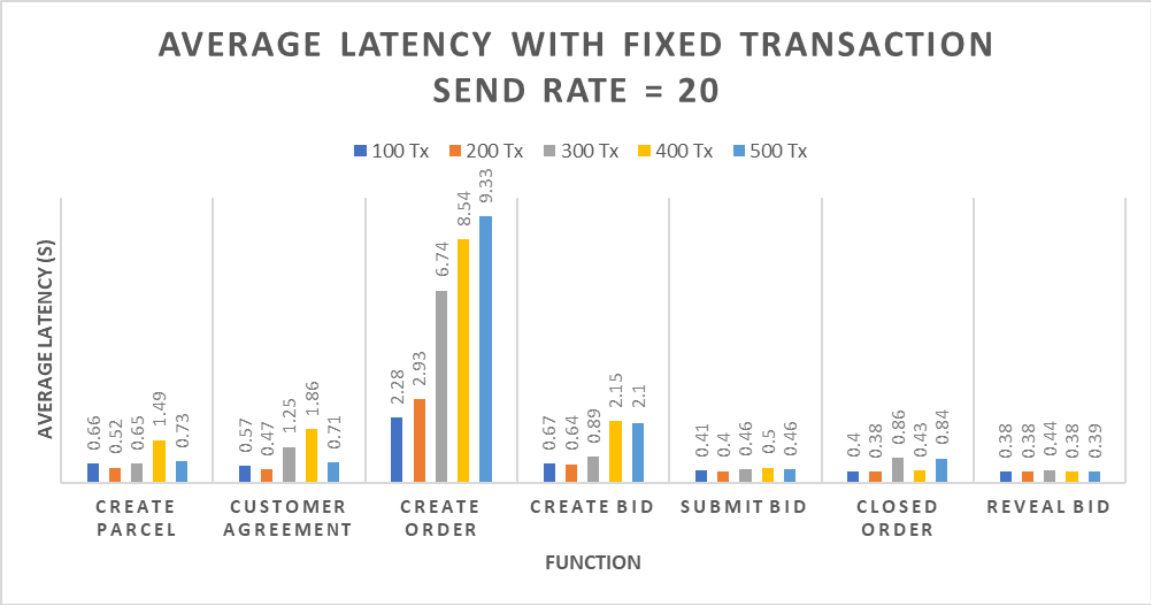

Fig. 6.4.5: Experiment I - Latency with TPS = 20

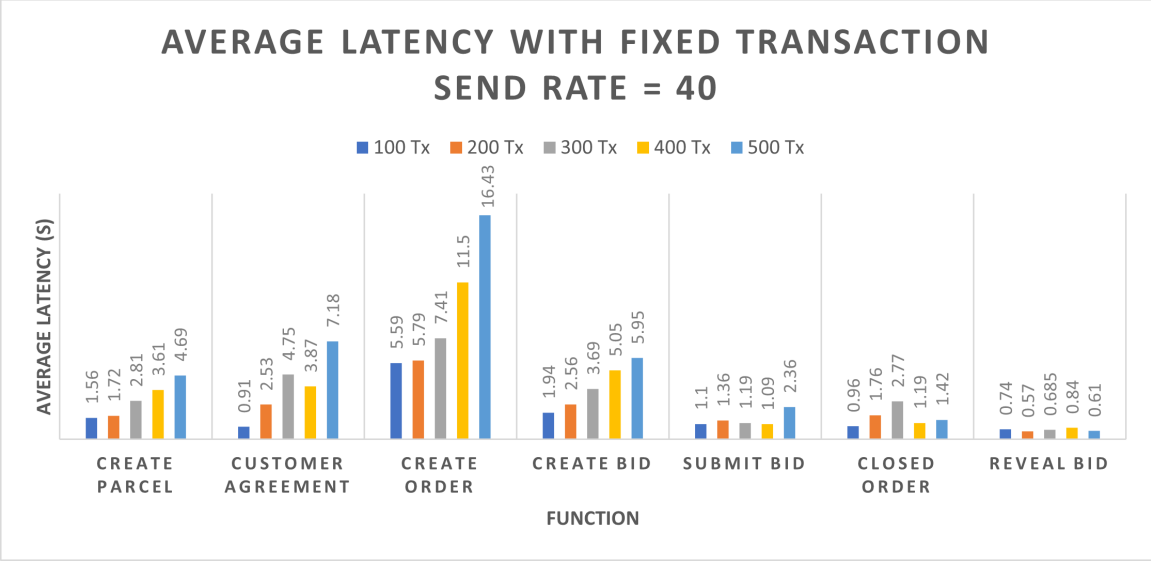

Fig. 6.4.6: Experiment II - Latency with TPS = 40

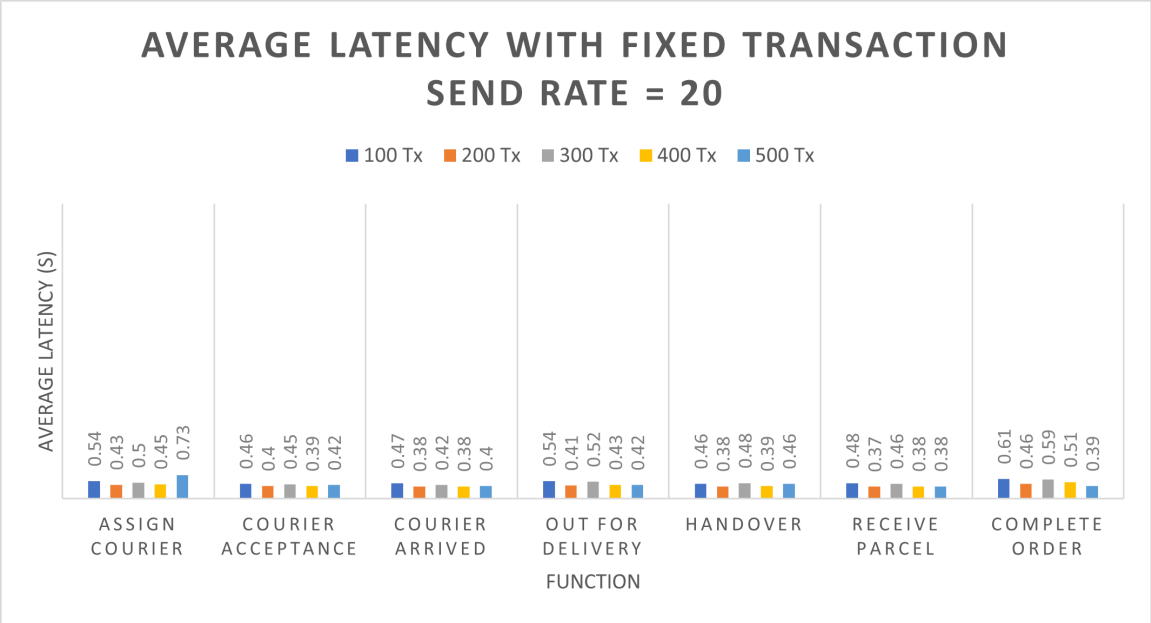

Fig. 6.4.7: Experiment I - Latency with TPS = 20

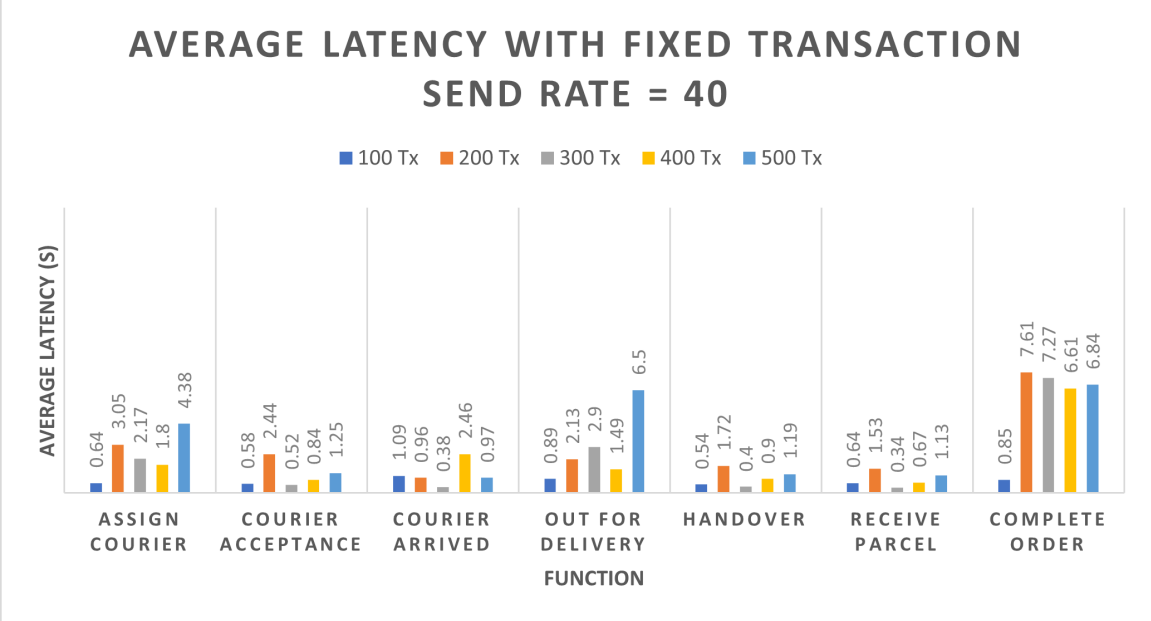

Fig. 6.4.8: Experiment II - Latency with TPS = 40

6.4.1 Analysis of Result

The experimental analysis reveals that the first experiment (TPS =20) attended a 100% success rate, while the second experiment (TPS = 40) achieved at least a 98% rate of transactions. The first two transactions fail due to an endorsement failure happens. Since we run the three organizations on the same virtual machine and the Caliper tool switch between the function's execution rounds within 5 seconds, it might take more time to switch between peers authorized to invoke this function automatically.

Figures 6.4.1 and 6.4.3 illustrate the transaction throughput with TPS equals 20. While figures 6.4.2 and 6.4.4 demonstrate the throughput when the transaction sent rate is 40 after executing chaincode's functions using 100 to 500 simultaneous transactions. It is observed that the throughput at a transaction send rate of 40 is slightly higher than when it is 20. Also, when the TPS is 20, it shows consistency in the throughput, which reflects the reliability and availability of Hyperledger. The throughput ranges between 17-19, except for the Create Order function. Create Order function has less throughput as it includes one read operation and two write operations (one on the ledger and the other on private data collection).

When the TPS is 40, the throughput is approximately similar for each function in figure 6.4.2, disregarding the transaction load. However, the throughput of functions listed in figure 6.4.4 significantly rises, especially when the transaction load is more than 300 transaction. The throughput is almost 1.5 times the throughput when TPS is 20. This growth refers to the nature of these functions, as they only update the state of the order and parcel. Besides, it doesn't write any data on world state, actual data are written on private data collection while only hashes are recorded on other organizations private collection. The maximum achieved throughput when TPS is 20 and 40 are 19.9 and 35.55 transaction, respectively.

Regarding latency, figures 6.4.5-6.4.8 describe the latency after executing the chaincode's functions, using 100 to 500 simultaneous transactions. It is noticed that when TPS is 20, the average latency follows a particular pattern and remains con-

sistent, particularly in the second batch of the functions shown in figure 6.4.7. The Create Order achieved the highest average latency, 9.33 seconds when TPS is 20 and 16.43 seconds when TPS is 40 as Create Order transactions require more time to be executed and written successfully on world state and private data collection. Furthermore, there is a continuous growth in the average latency as the number of transactions increases when TPS is 40. The reason behind that the transactions which are waiting at the orderer node are considerably growing each second. We can observe that the Complete Order function experience a rise in the average latency [7.61, 7.27, 6.61, 6.84] seconds when the transactions load is 200, 300, 400, and 500, respectively. This represents about five times the average latency of the same function when TPS is 20, which recorded [0.46, 0.59, 0.51, 0.39] seconds. This fall cause is that this function computes and updates the courier reputation and consumes additional time to commit the transaction outcome. Conversely, the Complete Order function's throughput has decreased compared to other functions in the same batch.

CHAPTER 7

Conclusion and Future Work

The work presented in this thesis provided a blockchain-based crowdshipping system that facilitates the delivery of parcels in a decentralized way and eliminates the need for a central authority or third-party involvement. Blockchain offers immutability by preventing tampering with the block's information that is stored throughout the chain. The data used daily by numerous business stakeholders is made more trustworthy and authentic by immutability. At the same time, crowd-sourcing the shipping enables faster service, such as same-day delivery.

This new architecture has been evaluated first using the STRIDE threat modeling framework so we can identify potential threats and suggest proper countermeasures. The research solves the scalability issue that exists in previous works, which require collateral from the system's participants to ensure they will only act honestly, making them impractical and irrelevant for a crowd-sourced solution. Therefore, we developed a reputation management algorithm that addresses the trust issues associated with participants' behavior and encourages honest conduct to aid the sellers in selecting trustworthy couriers to fulfill their requests confidently. Our design uses a permissioned blockchain to track interactions among participants and dynamically assign couriers' reputation scores based on these interactions.

The study conducts a prototype implementation of the proposed LMD platform. The full code of the chaincode with instructios to set up the network have been publicly made available on Github [4]. Besides, analysis concerning threats in reputation systems. Finally, performance measurement was performed on the application to measure the throughput and average latency.

The operational cost associated with the proposed solution was a critical factor in choosing the type of blockchain. For instance, V. Malhotra in [52] proposed a POD schema using Ethereum Blockchain. His study results show that using a cryptocurrency-based blockchain consumes substantial computational resources, as the shipping cost of a parcel could cost more than $400, making it an expensive option. Therefore, this thesis avoids this type of blockchain and costly consensus protocols to develop a feasible and practical solution.

In Hyperledger Fabric, there is no notion of cryptocurrency, which is the fuel for the costly proof of work consensus algorithm. The Hyperledger Fabric is entirely free and open-source. The only expense of the proposed platform is running up the network using interested parties' hardware or through a Cloud provider such as IBM, Amazon, etc. The cloud hosting option cost depends on the number of peer nodes, peer node storage, data written to the network, and data transfer. For example, a managed Hyperledger Fabric network by IBM costs approximately $875 per month [83]. Requiring couriers and sellers for annual membership fees could cover these charges.

Since blockchain technology is still in the early to middle stages of research and development, many unexplored areas could be studied. This research has limitations because sellers must wait for couriers to reveal their bids to assign a candidate courier. Future work might consider the automatic execution of this step. Besides, only allow a courier to submit multiple bids if there is no overlapping in the shipping requests time. Furthermore, we can add IoT sensors that send events to the system to automatically confirm courier arrival at the pickup location and drop-off destination. Off-chain computation for the reputation could be adopted to decrease the computational burden on the network if applied to a large-scale environment. Besides, future work might include studying how to set multiple packages to the single courier to make it cost-effective.

Another system drawback is that a validator peer in Hyperledger Fabric requires considerable computational resources to verify both the Block signature and the endorsement policy compliance of each transaction. So we may consider another

blockchain framework to develop the system in future work. A performance evaluation was carried out on a local computer with constrained CPU resources. As a result, when a high send TPS rate was used, the validator's capacity to execute quick validations was constrained. Thus, future work should apply a technique to deploy the application on a Cloud with high computational resource capabilities with extending the number of Hyperledger Fabric peers.

REFERENCES

[1] Aggarwal, D., Brennen, G., Lee, T., Santha, M., and Tomamichel, M. (2018). Quantum attacks on bitcoin, and how to protect against them. *Ledger*, 3.

[2] Alangot, B., Reijsbergen, D., Venugopalan, S., and Szalachowski, P. (2020). Decentralized lightweight detection of eclipse attacks on bitcoin clients. *2020 IEEE International Conference on Blockchain (Blockchain)*.

[3] Alnaggar, A., Gzara, F., and Bookbinder, J. H. (2021). Crowdsourced delivery: A review of platforms and academic literature. *Omega*, 98:102139.

[4] Alqaisi, A. (2022). Windsor advanced security & privacy labs. `https://github.com/WASPLab/`.

[5] AlTawy, R., ElSheikh, M., Youssef, A. M., and Gong, G. (2017). Lelantos: A blockchain-based anonymous physical delivery system. *2017 15th Annual Conference on Privacy, Security and Trust (PST)*.

[6] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A. D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K. A., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S. W., and Yellick, J. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. *Proceedings of the Thirteenth EuroSys Conference*.

[7] Archetti, C., Savelsbergh, M., and Speranza, M. G. (2016). The vehicle rout-

ing problem with occasional drivers. *European Journal of Operational Research*, 254(2):472–480.

[8] Arntz, P. (2019). Explained: War shipping: Malwarebytes labs. `https://blog.malwarebytes.com/explained/2019/10/explained-war-shipping`.

[9] Arslan, A., Agatz, N., Kroon, L. G., and Zuidwijk, R. A. (2016). Crowdsourced delivery – a pickup and delivery problem with ad-hoc drivers. *SSRN Electronic Journal*.

[10] Austin, R. (2017). Ride-austin-june6-april13 - dataset by ride-austin. `https://data.world/ride-austin/ride-austin-june-6-april-13`.

[11] Boudrez, F. (2007). Digital signatures and electronic records. *Archival Science*, 7(2):179–193.

[12] Boyen, X., Herath, U., McKague, M., and Stebila, D. (2021). Associative blockchain for decentralized pki transparency. *Cryptography*, 5(2).

[13] Brakeville, S. and Perepa, B. (2018). Blockchain basics: Introduction to distributed ledgers. `https://developer.ibm.com/tutorials/cl-blockchain-basics-intro-bluemix-trs/`.

[14] Breen, K. (2019). Food delivery app doordash reports data breach affecting 4.9m users - national. `https://globalnews.ca/news/5957123/doordash-data-breach/`.

[15] Cachin, C. (2016). Architecture of the hyperledger blockchain fabric.

[16] Casino, F., Dasaklis, T. K., and Patsakis, C. (2019). A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics*, 36:55–81.

[17] Castro, M. and Liskov, B. (2002). Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

[18] Chiluka, N., Andrade, N., Gkorou, D., and Pouwelse, J. (2012). Personalizing eigentrust in the face of communities and centrality attack. *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, page 503–510.

[19] Demir, M., Turetken, O., and Ferwom, A. (2019). Blockchain and iot for delivery assurance on supply chain (bidas). In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5213–5222.

[20] Dolan, S. (2022). Last mile delivery logistics explained: Problems & solutions. https://www.insiderintelligence.com/insights/last-mile-delivery-shipping-explained/.

[21] DoorDash (2021). Dasher ratings explained. https://help.doordash.com/dashers/s/article/Dasher-Ratings-Explained?language=en_US.

[22] EITZMAN, R. and DUNIN-UNDERWOOD, A. (2019). Cryptocurrency and blockchain networks: Facing new security paradigms. https://www.mandiant.com/resources/blog/cryptocurrency-blockchain-networks-facing-new-security-paradigms.

[23] Erko, A. (2022). Blockchain attack vectors: Vulnerabilities of the most secure technology. https://www.apriorit.com/dev-blog/578-blockchain-attack-vectors.

[24] Evans, P. (2020). Food delivery apps cut some restaurant fees amid surging demand due to covid-19 — cbc news. https://www.cbc.ca/news/business/food-delivery-apps-fees-1.5765790.

[25] FINTRAC (2021). Government of canada. https://fintrac-canafe.canada.ca/guidance-directives/client-clientele/Guide11/11-eng.

[26] Gaiha, V. G. V. G. G. (2020). Building a transparent supply chain. https://hbr.org/2020/05/building-a-transparent-supply-chain.

[27] Gambetta, D. (2000). Can we trust trust.

[28] Gao, Y.-L., Chen, X., Chen, Y.-L., Sun, Y., Niu, X., and Yang, Y.-X. (2018). A secure cryptocurrency scheme based on post-quantum blockchain. *IEEE Access*, 6:27205–27213.

[29] Gatta, V., Marcucci, E., Nigro, M., Patella, S., and Serafini, S. (2018). Public transport-based crowdshipping for sustainable city logistics: Assessing economic and environmental impacts. *Sustainability*, 11(1):145.

[30] Gatta, V., Marcucci, E., Nigro, M., Patella, S. M., and Serafini, S. (2019). Public transport-based crowdshipping for sustainable city logistics: Assessing economic and environmental impacts. *Sustainability*, 11(1).

[31] Geroni, D. (2022). Hybrid blockchain: The best of both worlds. `https://101blockchains.com/hybrid-blockchain/`.

[32] Goldberg, J. (2022). E-commerce sales grew 50% to $870 billion during the pandemic. `https://www.forbes.com/sites/jasongoldberg/2022/02/18/e-commerce-sales-grew-50-to-870-billion-during-the-pandemic/?sh=793fc4344e83`.

[33] Gonzalez, L. (2019). Blockchain, herding and trust in peer-to-peer lending. *Managerial Finance*, 46(6):815–831.

[34] Gurley, L. and Franceschi-Bicchierai, L. (2021). Hacking ring allegedly stole americans' identities, rented them to gig workers. `https://www.vice.com/en/article/wx5dv9/hacking-ring-allegedly-stole-americans-identities-rented-them-to-gig-workers`.

[35] Ha, X. S., Le, H. T., Metoui, N., and Duong-Trung, N. (2020). Dem-cod: Novel access-control-based cash on delivery mechanism for decentralized marketplace. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 71–78.

[36] Hameed, K., Barika, M., Garg, S., Amin, M. B., and Kang, B. (2022). A taxonomy study on securing blockchain-based industrial applications: An overview, application perspectives, requirements, attacks, countermeasures, and open issues. *Journal of Industrial Information Integration*, 26:100312.

[37] Hasan, H. R. and Salah, K. (2018). Blockchain-based solution for proof of delivery of physical assets. *Lecture Notes in Computer Science*, page 139–152.

[38] Henderson, C. (2022). Package delivery! cybercriminals at your doorstep. `https://securityintelligence.com/posts/package-delivery-cybercriminals-at-your-doorstep/`.

[39] Homoliak, I., Venugopalan, S., Reijsbergen, D., Hum, Q., Schumi, R., and Szalachowski, P. (2021). The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses. *IEEE Communications Surveys & Tutorials*, 23(1):341–390.

[40] Hyperledger (2018). Measuring blockchain performance with hyperledger caliper. `https://www.hyperledger.org/blog/2018/03/19/measuring-blockchain-performance-with-hyperledger-caliper`.

[41] Jones, D. (2018). How to secure 'permissioned' blockchains. `https://www.darkreading.com/endpoint/how-to-secure-permissioned-blockchains`.

[42] Kakei, S., Shiraishi, Y., Mohri, M., Nakamura, T., Hashimoto, M., and Saito, S. (2020). Cross-certification towards distributed authentication infrastructure: A case of hyperledger fabric. *IEEE Access*, 8:135742–135757.

[43] Kaur, R. (2021). What is the difference between smart contracts and dapps? `https://medium.datadriveninvestor.com/what-is-the-difference-between-smart-contracts-and-dapps-d252d88d32d3`.

[44] Kiktenko, E. O., Pozhar, N. O., Anufriev, M. N., Trushechkin, A. S., Yunusov, R. R., Kurochkin, Y. V., Lvovsky, A. I., and Fedorov, A. K. (2018). Quantum-secured blockchain. *Quantum Science and Technology*, 3(3):035004.

[45] Koutrouli, E. and Tsalgatidou, A. (2012). Taxonomy of attacks and defense mechanisms in p2p reputation systems—lessons for reputation system designers. *Computer Science Review*, 6(2):47–70.

[46] Lantz, L. and Cawrey, D. (2021). *Mastering blockchain: Unlocking the power of Cryptocurrencies, smart contracts, and Decentralized Applications*. O'Reilly.

[47] Le, C. (2022). Canada - ecommerce. `https://www.trade.gov/country-commercial-guides/canada-ecommerce`.

[48] Li, C., Chen, X., Chen, Y.-L., Hou, Y., and Li, J. Y. (2019a). A new lattice-based signature scheme in post-quantum blockchain network. *IEEE Access*, 7:2026–2033.

[49] Li, S., Wu, W., Xia, Y., Zhang, M., Wang, S., and Douglas, M. A. (2019b). How do crowd logistics platforms create value? an exploratory case study from china. *International Journal of Logistics Research and Applications*, 22(5):501–518.

[50] Liang, X., Shetty, S., Tosh, D. K., Foytik, P., and Zhang, L. (2017). Towards a trusted and privacy preserving membership service in distributed ledger using intel software guard extensions. In Qing, S., Mitchell, C., Chen, L., and Liu, D., editors, *Information and Communications Security - 19th International Conference, ICICS 2017, Beijing, China, December 6-8, 2017, Proceedings*, volume 10631 of *Lecture Notes in Computer Science*, pages 304–310. Springer.

[51] Luu, L., Chu, D.-H., Olickel, H., Saxena, P., and Hobor, A. (2016). Making smart contracts smarter. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

[52] Malhotra, V. (2021). A secure proof of delivery scheme for crowdsourced last mile delivery using blockchain.

[53] Malik, S., Dedeoglu, V., Kanhere, S. S., and Jurdak, R. (2019). Trustchain: Trust management in blockchain and iot supported supply chains. *2019 IEEE International Conference on Blockchain (Blockchain)*.

[54] Marcus, Y., Heilman, E., and Goldberg, S. (2018). Low-resource eclipse attacks on ethereum's peer-to-peer network. *IACR Cryptol. ePrint Arch.*, 2018:236.

[55] Mastilak, L., Helebrandt, P., Galinski, M., and Kotuliak, I. (2022). Secure inter-domain routing based on blockchain: A comprehensive survey. *Sensors*, 22(4):1437.

[56] Midtlyng, J. (2022). Food delivery apps [detecting location spoofing fraud]. `https://www.incognia.com/blog/food-delivery-app-fraud-location-spoofing`.

[57] Morganti, G., Schiavone, E., and Bondavalli, A. (2018). Risk assessment of blockchain technology. In *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*, pages 87–96.

[58] N, B. (2020). Hackers leaked ubereats data on darkweb. `https://cybersecuritynews.com/hackers-leaked-ubereats-data-on-darkweb/`.

[59] Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*.

[60] Ngamsuriyaroj, S., Likittheerameth, T., Kahutson, A., and Pathummasut, T. (2018). Package delivery system based on blockchain infrastructure. In *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, pages 1–6.

[61] Oligeri, G., Sciancalepore, S., Ibrahim, O. A., and Di Pietro, R. (2019). Drive me not: Gps spoofing detection via cellular network: (architectures, models, and experiments). In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '19, page 12–22, New York, NY, USA. Association for Computing Machinery.

[62] Ontario, G. o. (2021). Digital id in ontario. `https://www.ontario.ca/page/digital-id-ontario`.

[63] Placek, M. (2022). Couriers and local delivery services market size in canada 2018-2021. `https://www.statista.com/statistics/1156206/couriers-and-local-delivery-services-market-size-canada/`.

[64] Pourrahmani, E. and Jaller, M. (2021). Crowdshipping in last mile deliveries: Operational challenges and research opportunities. *Socio-Economic Planning Sciences*, 78:101063.

[65] Putz, B. and Pernul, G. (2020). Detecting blockchain security threats. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 313–320.

[66] Putz, B. and Pernul, G. (2022). *Trust Factors and Insider Threats in Permissioned Distributed Ledgers*, page 25–50. Springer-Verlag, Berlin, Heidelberg.

[67] Qi, W., Li, L., Liu, S., and Shen, Z.-J. M. (2018). Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact. *Manufacturing & Service Operations Management*, 20(4):737–751.

[68] Saad, M., Spaulding, J., Njilla, L. L., Kamhoua, C. A., Shetty, S. S., Nyang, D., and Mohaisen, A. (2019). Exploring the attack surface of blockchain: A systematic overview. *ArXiv*, abs/1904.03487.

[69] Shahda, W. (2022). Protect your solidity smart contracts from reentrancy attacks. `https://medium.com/coinmonks/protect-your-solidity-smart-contracts-from-reentrancy-attacks-9972c3af7c21`.

[70] Shevchenko, N., Chick, T. A., O'Riordan, P., Scanlon, T. P., and Woody, C. (2018). Threat modeling: A summary of available methods.

[71] Shostack, A. (2014). *Threat modeling: Designing for security.* Wiley.

[72] Sifra, E. M. (2022). Security vulnerabilities and countermeasures of smart contracts: A survey. *2022 IEEE International Conference on Blockchain (Blockchain).*

[73] Simmons, G. J. (1979). Symmetric and asymmetric encryption. *ACM Computing Surveys*, 11(4):305–330.

[74] Squarepants, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *SSRN Electronic Journal.*

[75] Subramanian, G. and Sreekantan Thampy, A. (2021). Implementation of blockchain consortium to prioritize diabetes patients' healthcare in pandemic situations. *IEEE Access*, 9:162459–162475.

[76] Sun, Y., Apostolaki, M., Birge-Lee, H., Vanbever, L., Rexford, J., Chiang, M., and Mittal, P. (2021). Securing internet applications from routing attacks. *Communications of the ACM*, 64(6):86–96.

[77] Supreet, Y., Vasudev, P., Pavitra, H., Naravani, M., and Narayan, D. (2020). Performance evaluation of consensus algorithms in private blockchain networks. In *2020 International Conference on Advances in Computing, Communication & Materials (ICACCM)*, pages 449–453.

[78] Tagiltseva, J. A., Kuzina, E. L., Drozdov, N. A., Vasilenko, M. A., Kuzina, M. A., and Korenyakina, N. N. (2021). The international cargo transportations in south russian transport and logistics system. In *2021 International Conference on Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS)*, pages 248–251.

[79] Taylor, R. (2022). What is dnssec and how does it work? `https://bluecatnetworks.com/blog/breaking-down-dnssec-how-does-it-work/`.

[80] Team, I. (2018). Eclipse attacks on blockchains' peer-to-peer network. `https://medium.com/hackernoon/eclipse-attacks-on-blockchains-peer-to-peer-network-26a62f85f11`.

[81] Technavio (2022). North america last mile delivery market by service and vehicle type - forecast and analysis 2023-2027. `https://www.technavio.com/report/last-mile-delivery-market-size-in-north-america-industry-analysis`.

[82] Teksaz (2020). Doordash customer ratings manipulation continues... `https://www.uberpeople.net/threads/doordash-customer-ratings-manipulation-continues.376921/`.

[83] Tiwari, A. (2019). Where to deploy your hyperledger fabric network and why? `https://medium.com/@ayushtiwari_87831/where-to-deploy-your-hyperledger-fabric-network-and-why-7a4377a66009`.

[84] Truong, N., Lee, G. M., Sun, K., Guitton, F., and Guo, Y. (2021). A blockchain-based trust system for decentralised applications: When trustless needs trust. `https://arxiv.org/abs/2101.10920`.

[85] Uber (2021). How ratings work for delivery people — uber. `https://www.uber.com/us/en/deliver/basics/tips-for-success/delivery-ratings-explained/`.

[86] Van Meter, R. and Horsman, D. (2013). A blueprint for building a quantum computer. *Commun. ACM*, 56(10):84–93.

[87] Wang, M. (2021). I tried delivering parcels for a day. `https://www.linkedin.com/pulse/i-tried-delivering-parcels-day-mark-wang`.

[88] Wang, S., Tang, X., Zhang, Y., and Chen, J. (2019). Auditable protocols for fair payment and physical asset delivery based on smart contracts. *IEEE Access*, 7:109439–109453.

[89] Xiong, L. and Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(07):843–857.

[90] Xu, G., Guo, B., Su, C., Zheng, X., Liang, K., Wong, D. S., and Wang, H. (2020). Am i eclipsed? a smart detector of eclipse attacks for ethereum. *Computers & Security*, 88:101604.

[91] Zhang, S. and Lee, J.-H. (2020). Analysis of the main consensus protocols of blockchain. *ICT Express*, 6(2):93–97.

[92] Zhou, Z., Wang, M., Yang, C.-N., Fu, Z., Sun, X., and Wu, Q. J. (2021). Blockchain-based decentralized reputation system in e-commerce environment. *Future Gener. Comput. Syst.*, 124(C):155–167.

VITA AUCTORIS

NAME:	Ala' Alqaisi
PLACE OF BIRTH:	Amman, Jordan
YEAR OF BIRTH:	1991
EDUCATION:	Amman Arab University, Bachelor of Computer Information Systems (CIS), Jordan, 2012
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2023