

Wilfrid Laurier University

Scholars Commons @ Laurier

Theses and Dissertations (Comprehensive)

2019

Towards Secure and Fair IIoT-Enabled Supply Chain Management via Blockchain-based Smart Contracts

Amal Eid Alahmadi

Wilfrid Laurier University, alah6650@mylaurier.ca

Follow this and additional works at: <https://scholars.wlu.ca/etd>



Part of the [Information Security Commons](#), [Other Computer Sciences Commons](#), and the [Programming Languages and Compilers Commons](#)

Recommended Citation

Alahmadi, Amal Eid, "Towards Secure and Fair IIoT-Enabled Supply Chain Management via Blockchain-based Smart Contracts" (2019). *Theses and Dissertations (Comprehensive)*. 2147.
<https://scholars.wlu.ca/etd/2147>

This Thesis is brought to you for free and open access by Scholars Commons @ Laurier. It has been accepted for inclusion in Theses and Dissertations (Comprehensive) by an authorized administrator of Scholars Commons @ Laurier. For more information, please contact scholarscommons@wlu.ca.

Towards Secure and Fair IIoT-Enabled Supply Chain Management via Blockchain-based Smart Contracts

by

Amal Alahmadi

THESIS

Submitted to the Department of Physics and Computer Science

Faculty of Science

in partial fulfilment of the requirements for

Degree of Master of Applied Computing

Wilfrid Laurier University

April 2019

Copyright ©Amal Alahamdi, 2019

Abstract

Integrating the Industrial Internet of Things (IIoT) into supply chain management enables flexible and efficient on-demand exchange of goods between merchants and suppliers. However, realizing a fair and transparent supply chain system remains a very challenging issue due to the lack of mutual trust among the suppliers and merchants. Furthermore, the current system often lacks the ability to transmit trade information to all participants in a timely manner, which is the most important element in supply chain management for the effective supply of goods between suppliers and the merchants. This thesis presents a blockchain-based supply chain management system in the IIoT. The proposed system takes advantage of blockchain technology in terms of its transparency and tamper-proof nature to support fair goods exchange between merchants and suppliers. Additionally, the decentralization and pseudonymity property will play a significant role in preserving the privacy of participants in the blockchain. In particular, fairness in the IIoT is first defined. Then, a design for a smart contract for fair goods exchange is presented to prevent malicious behavior through imposing penalties. The proposed system was prototyped on Ethereum and experiments were conducted to demonstrate its feasibility.

This is for you, Dad.

Even from across the sea, I can feel your love

Acknowledgements

Arriving at this stage and writing this acknowledgement was always a dream I hoped to achieve one day. In my journey towards this degree, I met many people who I have to credit for helping me to arrive at this stage, otherwise this dream would not have been achieved.

I will start with an inspiration, a teacher and brother, Dr. Xiaodong Lin, who was a pillar of support and a role model for me throughout my journey. He always believed in me and gave me this opportunity. Furthermore, he has always been there at all times providing his heartfelt help and support in addition to giving me invaluable suggestions and guidance in my quest for knowledge. I shall be eternally grateful to Dr. Xiaodong Lin for his assistance.

I take great pleasure extending my appreciation to my colleagues in the BBCR Group at the University of Waterloo, for all their help and support. Special thanks to Li Ming , Dongxiao Liu , Yuan Zhang and Anjia Yang whose precious friendship I will always cherish.

Special thanks also go to Dr. Ilias Kotsireas and my examining committee members for the time and effort they have given, and to all my friends at Durham College, especially prof. Karl Alexander for his support.

Last but not least, I would like to thank all my family who provided me with unconditional love during my academic journey here in Canada.

Amal

Contents

List of Tables	iii
List of Figures	iv
1 Introduction	1
1.1 Overview	1
1.2 Motivation	2
1.3 Objectives	3
1.4 Thesis Outline	3
2 Background and Literature Review	5
2.1 Blockchain Technology	5
2.1.1 Concept of Blockchain	5
2.1.2 Decentralized Networks and Ledgers	7
2.1.3 Blockchain Architecture	7
2.1.4 Consensus Function	9
2.1.5 Smart Contract: Ethereum	10
2.1.6 Security and privacy	12
2.1.6.1 Cryptographic techniques	13
2.1.6.2 Privacy enhancing techniques	15
2.2 Supply chain system	17
2.2.1 Digital supply chain (DSC)	18
3 Secure and Fair IIoT-Enabled Supply Chain Management via Blockchain-based Smart Contracts	21
3.1 Problem formulation	21
3.1.1 System model	21
3.1.2 Threat model	23
3.1.3 Design goals	23
3.2 Proposed Scheme	24

3.2.1	The System architecture	24
3.2.2	The Scheme	25
3.2.2.1	Initialize Phase	26
3.2.2.2	Order Phase	27
3.2.2.3	Confirmation Phase	29
3.2.2.4	Delivery Phase	30
3.2.2.5	Judgement Phase	30
3.3	Security analysis	32
4	Experiment Results	33
4.1	Experiment setup framework	33
4.1.1	Building a Parity Ethereum Client	34
4.1.1.1	Technical Tests	35
4.1.2	Building Proof-of-Authority Chains	37
4.1.2.1	Technical Tests	43
4.1.3	Smart contract	44
4.2	Performance evaluation	45
4.2.1	Private network	45
4.2.2	Public network	45
5	Conclusions and Future Works	48
5.1	Summary of research	48
5.2	Future research directions	48
A	Codes	50
	Bibliography	56

List of Tables

2.1	Consensus protocols comparison	10
3.1	Notation explanation	27

List of Figures

2.1	A sequence of blocks	5
2.2	Blockchain Architecture	8
2.3	Smart Contract	11
2.4	How transactions are verified and chained together	14
3.1	System Model	22
3.2	System Architecture	25
3.3	Outline of fair goods exchange	26
4.1	Structure of the Parity Ethereum Client	33
4.2	Connecting with the official public blockchain	35
4.3	Connected with web3	36
4.4	Chain configuration	38
4.5	Resolving the chain	38
4.6	configure Node 0	39
4.7	configure Node 1	39
4.8	Authority address 1	39
4.9	User address	40
4.10	Authority address 2	40
4.11	Filed chain	41
4.12	Completed file Node 0	41
4.13	Completed file Node 1	42
4.14	Run node0	42
4.15	Run node1	42
4.16	Nodes are connected	44
4.17	Estimates the Gas versus the USD of each phase	46
4.18	Transaction confirmation time in Rinkeby	46

Chapter 1

Introduction

1.1 Overview

Supply chain management systems can coordinate cross-organization processes to move products from suppliers to merchants. Empowered by the emerging Industrial Internet of Things (IIoT) technology, modern supply chain management promises to provide industrial sectors with more efficient on-demand resource management services [35]. In a supply chain system, merchants and suppliers can dynamically exchange goods, which makes it increasingly important to enforce a reliable and affordable supply chain management system [40]. In particular, the demand for fairness requires merchants and suppliers to correctly follow their contracts and behave honestly [12]. This issue becomes more challenging when there is a lack of transparency and mutual trust in the current supply chain systems. According to the statistics [1], concern for ensuring an ethical supply chain has continued to proliferate since 2018. The emerging blockchain technology is regarded as a promising solution to the above issue [25] [16] [29]. Blockchain technology currently plays a significant role as a foundation for distributed ledgers that can offer an innovative platform for decentralized and transparent transaction mechanisms in industries [2]. When applied to a supply chain system, blockchain technology can significantly enhance transparency and credibility for fair goods exchange. As a result, large corporations and organizations are moving aggressively towards building a blockchain-based infrastructure while boosting

mutual trust among industrial entities [16].

However, realizing the promises of such a blockchain based system still faces non-trivial challenges. Firstly, there is no straightforward solution to designing a smart contract that preserves fairness. Secondly, the feasibility and implementation of such a system remains unsolved. This thesis presents a proposal for a fair and efficient supply chain management system based on blockchain. It allows suppliers and merchants to exchange goods and prevent malicious behaviors. The contributions of this study can be summarized as follows:

- To first provide a definition of the fairness requirements for supply chain management, followed by extending an existing smart contract for fair digital goods exchange [15]. The smart contract is also designed in order to achieve fair goods exchange for IIoT-enabled supply chain management.
- By enforcing penalties, our designed smart contract can compel the involved parties to honestly fulfill their obligations. Moreover, this present work implements the smart contract on the Ethereum network and experimental results demonstrate the feasibility of the proposed scheme.

1.2 Motivation

Supply chain management (SCM) is considered as a sequence of main processes that span many industries on a global level, whereby it is the network that stands between an organization and its suppliers. These processes involve several phases where data flows are distributed in different directions, starting from manufacturers through suppliers and distributors, to customers. This illustrates the major importance of data flow, which can either support or impact critical business decisions. Although there is high demand for such a system, traditional centralized supply chain information systems lack many of the characteristics, such as security, transparency and fairness, that are considered as necessary

in real-time for many businesses. This thesis provides a conceptual theoretical framework based on a distributed solution of a private ledger to share data among trading partners in real-time and with full transparency and fairness. The proposal presented in this work uses a distributed ledger system that is based on blockchain technology that documents all exchange custody events related to commerce between parties. This system is considered under the private and open type of blockchain [5] [18] where these exchanges are distributed between trading partners by contract only, thereby providing security and privacy for participants.

1.3 Objectives

The aim of this research is to investigate a new approach to developing and examining a distributed system that can share trading transactions or information related to supply chain management in the case of buying and selling between parties while ensuring a commercial fairness protocol. This model depends on blockchain technology that is executed on the Ethereum framework without any third-party interference, censorship or chance of fraud. Furthermore, this present work performs a security analysis and conducts experiments to illustrate the feasibility and efficiency of the proposed scheme. Particular emphasis is placed on the realization of a fairness protocol in the smart contract. The concept of fairness property for each side is defined, together with a description of the process needed to achieve fairness in this scheme.

1.4 Thesis Outline

The comprehensive structure of this thesis takes the form of five chapters, including this introductory chapter. Chapter 2 begins by laying out the background to blockchain technology and the supply chain, with a summary of related work in the field. Chapter 3 is concerned with the main proposal of the thesis, which is secure and fair IIoT-enabled

supply chain management via blockchain-based smart contracts, including the problem formulation used for this study, and presents the proposed research scheme, focusing on the system architecture and scheme model. Chapter 4 presents the implementation and analyzes the results of the data model using the Ethereum of the proposed framework. The final chapter, which is the conclusion, gives a brief summary with a critique of the findings and areas for further research in this field are identified.

Chapter 2

Background and Literature Review

2.1 Blockchain Technology

2.1.1 Concept of Blockchain

A blockchain is an immutable record or ledger which, as a growing chain, contains many blocks that link together using a cryptographic hash. By way of explanation, similar to a conventional public ledger, blockchain refers to the arrangement of a sequence of blocks in which a comprehensive record of a complete list of transactions records is contained [13]. Each block in a chain includes a cryptographic hash that is typically like a point of the immediately previous block. Transaction data and a timestamp are also included.

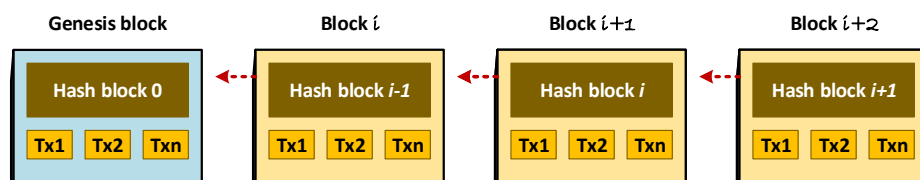


Figure 2.1: A sequence of blocks

Connection is made through a hash value that serves as a reference, known as the parent block [8]. The genesis block has no parent block since it is the first block of a blockchain, as

shown in Fig. 2.1. Blockchain started from a White Paper published in 2008 by an anonymous author using the name Satoshi Nakamoto [36]. The concept consists of a peer-to-peer version of digital currency that allows cash to be sent directly from one party to another online party without going through a third party, as is the norm in financial institutions. The critical concepts of blockchain include group consensus, trustless-ness, immutability, decentralization, distributed ledgers and security. At the core of blockchain is a ledger, which can be defined as a record-keeping device that allows keepers of that record to tell a story in addition to knowing and tracking asset ownership or any type of imaginable data. Although blockchain is considered as a new and cutting-edge technology, it is in fact a creative amalgamation of many old concepts of methodologies and technologies. The main attributes of blockchain include:

- **Decentralisation:** The validation of each transaction is a crucial element in conventional centralized systems. Unlike the validation performed by central trusted agencies, which often results in bottlenecks of performance and cost, blockchain network transactions can be achieved between any two peers (P2P), without authentication from a central agency. The concept results in a significant reduction in equipment costs in addition to overheads related to the development and operation of those systems, as well as a decrease in pressure and suffocating performance of the main server. Furthermore, this mechanized system prevents having a single point of failure or dependency in the case where, when any node comes back online after being offline for any reason, it can sync back to the current ledger with other online nodes.
- **Persistency:** It is almost impossible to tamper with transactions spread across a network since they must first be verified then recorded in blocks before they are distributed in the main ledger in the blockchain. Additionally, each transactions spread is checked while the broadcasted block is validated by other nodes. Thus, it is easy to detect any falsification.

- **Anonymity:** Each user can generate an address useful in interacting with the blockchain network. Different addresses can also be generated by users in instances where they want to avoid exposing their identity. In this way, no private user information is kept with any central entity. Additionally, a certain amount of privacy is preserved on the blockchain transactions. However, the intrinsic constraint nature of the blockchain cannot guarantee perfection in privacy preservation.
- **Auditability:** Validating and recording each blockchain transaction with a time-stamp makes it easy for users to verify and trace preceding records in the distributed network by accessing any node. The Bitcoin blockchain provides an iterative mechanism through which a transaction can be traced to the previous transaction, thus enhancing the transparency and traceability of the stored data.

2.1.2 Decentralized Networks and Ledgers

The blockchain ledger uses a peer network for updating and storage. The job of each node is to maintain its own ledger copy in the network, hence all the nodes come to a consensus regarding the contents of their updated ledger. This gives a guarantee that each individual copy of a ledger with nodes in the network is identical, which means that no centralized ledger is required. Moreover, if one of the nodes goes offline, the network continues to function seamlessly where the node can synchronize with the most recent ledger to be updated with the current state in the network. This allows for this technology to have no single point of failure and is the cause of the considerable interest in blockchain in parts of the world with developing infrastructures.

2.1.3 Blockchain Architecture

The main components of a block are the header and the body, as shown in Fig. 2.2. The specific components of a header are [48]:

- **Version:** This highlights the specific set of block validation rules to be followed.
- **PrevHash:** The hash value made up of 256 bits that link the current block to the previous block.
- **Timestamp:** The current timestamp as seconds since 1st January 1970 at Time 00:00 UTC.
- **nBits:** These present the hashing target in a compressed layout.
- **Nonce:** A field with only 4-byte size that often begins with 0. However, it goes up.
- **Merkle tree root:** This contains the hash value of all block transactions.

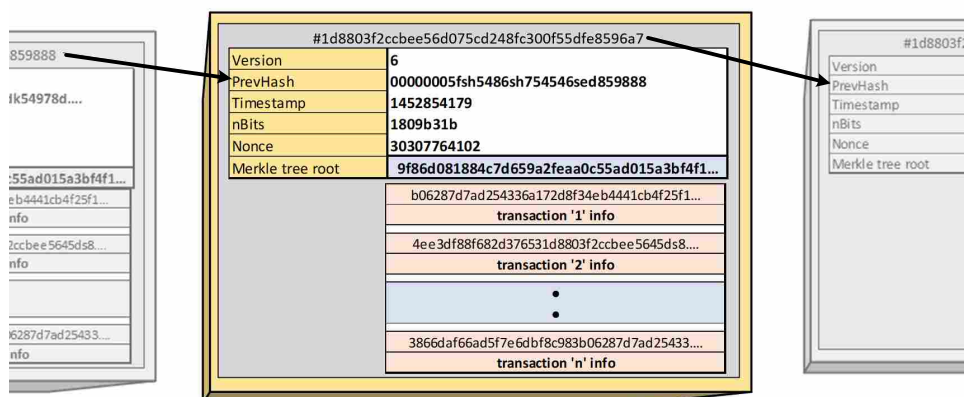


Figure 2.2: Blockchain Architecture

The other main component of the block is the body that is made up of a transaction counter, which tracks completed transactions. The maximum size limit of block transactions depends on the size of each transaction along with the size of the block. Authentication of transactions in the blockchain is processed using an asymmetric cryptography mechanism [48]. In the event of an untrustworthy environment, digital signatures based on asymmetric cryptographies are used.

The current blockchain can be categorized into three types, namely public, private and consortium, the selection of which often depends on a company's business [7]. With public

blockchain, everyone is able to verify and check a transaction as well as participate in the process of reaching consensus. As an example, Ethereum as a platform [9] and Bitcoin as an application are both public blockchain. However, in a private blockchain such as Hyperledger [9], the node will be more restricted and has authority management in terms of data access. For either consortium type, the platform can be open or closed, which means that the node with authority can be chosen in advance. This kind of blockchain is usually chosen in the case of partnerships, that is between two or more closed companies when they are considered as partly decentralized, such as the R3CEV Corda Platform [6]. Numerous public blockchains, which emerge on an almost daily basis, attract a multitude of users since they are open to the world. Many companies implement this blockchain for auditability and efficiency. Additionally, there is a consortium blockchain that supports a number of business applications. Currently, two of the best known platforms are Hyperledger and Ethereum. Hyperledger creates business consortium blockchain frameworks. The tools for building this consortium blockchain are provided by Ethereum. A comparison of the different types of blockchain is provided in Table 2.1.

2.1.4 Consensus Function

The case of the Byzantine Generals problem is helpful in reaching a consensus among the untrustworthy nodes [28]. In this case, a group of generals, who were tasked with commanding some of the Byzantine armies surrounding a city, were told that they must agree on whether or not to attack the city. Thus, if some of the generals were to attack the city, the attack would fail. However, among them there could be traitors who might create an untrustworthy environment by communicating different information to different generals, thus making it difficult to reach a consensus. The distributed nature of the blockchain network poses a similar challenge in terms of distributed nodes. Ledgers in all nodes in a blockchain must have the the same copy which, since the nodes are not centralized, is not achievable. Nodes must trust other nodes and this is facilitated by protocols that ensure

consistency among ledgers in different nodes. To highlights common approaches for reaching a consensus in the blockchain, there are many consensus algorithms mechanisms in a blockchain, which vary according to their purpose as well as their respective advantages and disadvantages. However, they all have to agree to the same block that ensures that the verified block has been correctly added to the chain in the blockchain. Table 2.1 provides a comparison between different consensus algorithms [42]. The comparison was made according to properties provided by [31] in terms of node identity management, which means that a node could be identified or could anonymously join a network. Energy saving is also included as some protocols require electricity to process, as is the tolerated power of the adversary, which refers to the hash power which is considered as the starting point for taking control of the network. Finally, the table includes some examples of applications or platforms that use these protocols. Of the many types of blockchain protocols, this present work focuses on the three that are best known.

Protocols	Node identity management	Energy saving	Tolerated power of adversary	Example
PoW	Open	No	<25% computing power	Bitcoin
PoS	Open	Partial	<51% stake	Peercoin
PBFT	Permissioned	Yes	<33.3% faulty replicas	Hyperledger Fabric
PoA	Permissioned	Yes	authority node only	Kovan and Parity

Table 2.1: Consensus protocols comparison

As shown in the above table, both PBFT and PoA are permissioned protocols, which means that the identity of a node wishing to join consensus must be known to the entire network. As a consequence, this applies more to commercial than public modes. In contrast, PoS and PoW are more appropriate for public blockchain.

2.1.5 Smart Contract: Ethereum

The smart contract, as one of the Turing-complete programming languages, was first proposed by Szabo [26]. In 2015, Ethereum introduced a new concept of blockchain 2.0

platforms by adding the idea of smart contracts that enable developers to add custom rules and logic to their transactions. This consequently changed the concept of blockchain, recognizing that it can do more than merely store data. Blockchain has now become a fully-fledged platform for application development, such that business processes can be automated and modelled on the one platform, which is the same as that used for their transaction data Fig 2.3. Enabling smart contracts means that blockchain can be applied to many scenarios [46]. Known as a chain code in some platforms, it can also be described as an

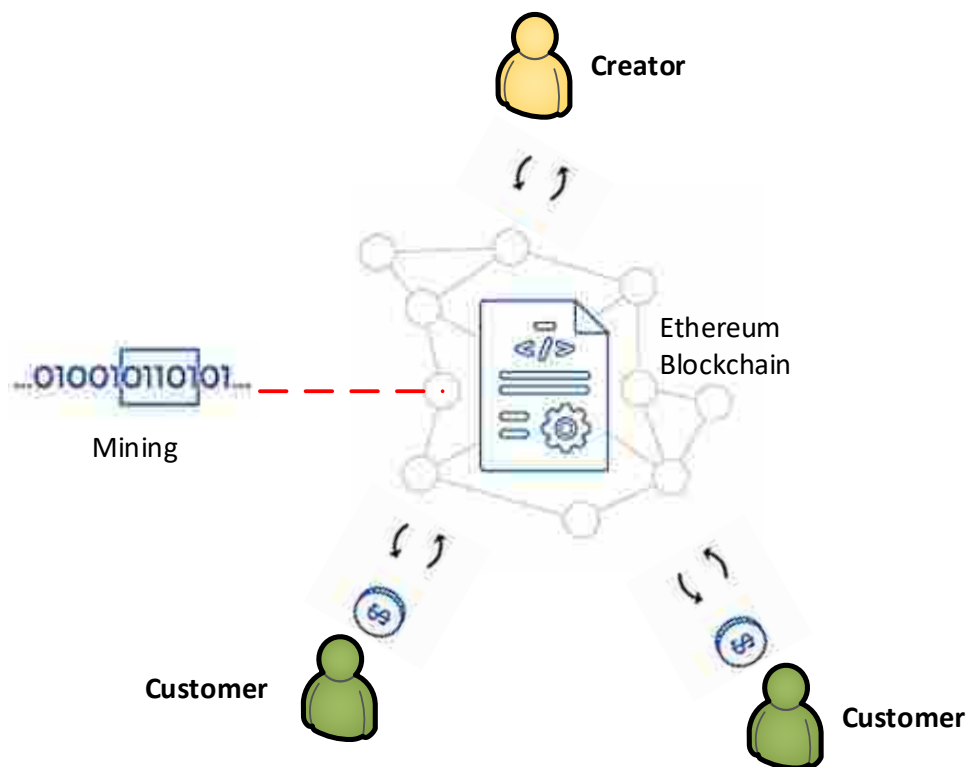


Figure 2.3: Smart Contract

electronic contract of a self-executed program with valid inputs of rules, conditions, precedents and decision points into transactions in the blockchain. The smart contract supports users in making a deposit on the blockchain that they cannot redeem before the unlock time [8]. This is vital for the proposed payment scheme that is introduced in Chapter 3. However, it is essentially a type of transaction in a blockchain by which each full node can verify execution of the process. Smart contracts have many advantages :

- **Autonomy:** Anyone can develop contracts, which means that there is no need for intermediaries such as auditors or lawyers.
- **Cost:** Dispensing with intermediaries often lowers costs.
- **Accuracy:** Since human intermediaries are being replaced with executable codes, the process will ensure the same performance without manipulation.
- **Backup:** There is a permanent record on the blockchain that allows for traceability or auditing, even if the author is no longer in business.

In Ethereum Blockchain, there is a concept called Gas, which is simply the amount paid by users as the cost for a transaction to be validated or processed on the blockchain. It is a reward separate from the consensus mining reward that is given to all miners independently, which means that it is a reward for all nodes on the network. Each transaction on the Ethereum blockchain must be submitted with Gas. However, it is important to understand that Gas is only consumed when data is written to the network whereas reading consumes no Gas. The cost of Gas varies from one contract to another as it directly depends on the type and complexity of the operation; more complex operations will cost more Gas than those that are simple. Consequently, architects and developers are keen to check their Gas costs on online Ethereum Gas Stations for an estimate based on the operations under development.

2.1.6 Security and privacy

From the perspective of security enhancement, the rapid increase in varieties of new mobile devices and service providers in recent years has highlighted vulnerability issues regarding malicious nodes. Various anti-malware filters have been offered to isolate malicious files that use the same matching pattern schemes. The detected virus patterns are stored and updated in a central server. Even so, malicious attackers still take advantage

of vulnerable centralized countermeasures. BitAV, a novel anti-malware environment, was proposed by Noyes [37] to help users spread malicious code on blockchain in a bid to enhance the security of distributed networks. According to Noyes [37], BitAV can enhance fault accuracy and scanning speed, which makes the network less vulnerable to directed denial-of-service attacks. The reliability of security infrastructure can also be improved using blockchain technologies. As an example, malicious attacks or flaws in hardware and software components contribute to a single point of failure exhibited in conventional public key infrastructures (PKIs). According to [3], a privacy-aware PKI can be constructed using blockchain while concurrently enhancing the reliability of conventional PKIs. Furthermore, privacy protection where mobile service and social network providers collect sensitive data about individuals adds to the increased risk with which users' private data is exposed to malware. Additionally, storing the information on central servers increases the susceptibility to malicious attacks. As an illustration, more than 300 petabytes of users' personal data have been collected by Facebook since it began [41]. Blockchain can enhance the privacy of sensitive information through a management system that decentralizes personal data, thus ensuring user ownership of private data is maintained [49]. Some of the privacy issues that the system protects include data ownership, audibility, transparency and gained access control.

2.1.6.1 Cryptographic techniques

Each entity in blockchain has a private key used for signing transactions and a public key that is used for the purpose of accessing the signed digital transactions that are published throughout the entire network. The characteristics of a digital signature encompass two phases, starting with the signing phase then the verification phase. As an example, a user known as Alice wants to send a transaction to Bob, another user. Alice first needs to sign on a transaction by generating a hash value that will be from the same transaction she will send. Next, she will encrypt that hash using her private key. The original data

contained in the encrypted hash is then sent to Bob. On the other side, Bob will verify the transaction by comparing the decrypted hash value using the public key owned by Alice with the hash value obtained from the data received by the hash function similar to Alice's. Blockchain will use a digital signature algorithm, which is an elliptic curve algorithm, to sign their transaction in the P2P network [49]. The elliptic curve digital signature algorithm (ECDSA) is considered as one type of digital signature algorithms. There is an approach by public-key cryptography that depends on the algebraic structure of elliptic curves that are applied to some limited fields. As a consequence, it requires smaller keys to provide equivalent security to that of non-EC cryptography.

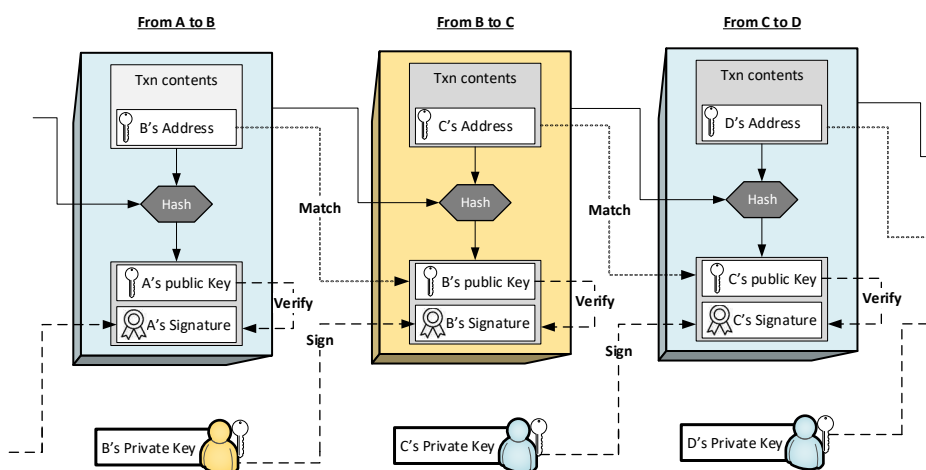


Figure 2.4: How transactions are verified and chained together

In a technically simplified example, Figure 2.4 gives a view of how the transaction on blockchain is signed, then linked together. Assuming that B wants to send bitcoins to C, the transaction will be hashed and signed with B's private key and the transaction will contain the hash of the previous transaction and B's public key. Anyone in the network can verify it as authorized by B. In the first place, B's public key has to match B's address according to the previous transaction, which will prove that the public key is valid. Subsequently, B's signature on the transaction will be verified by using B's public key. This phase ensures that the transaction is authorized and valid by sender B. However, B's public key is not

public until the key is used on a previous verification process. From that point, it can be understood that cryptocurrencies are passed from one address to another via a chain of transactions. Thus, each phase in the transaction chain can be checked to ensure that the bitcoins are spent validly.

2.1.6.2 Privacy enhancing techniques

The traditional blockchain technology does not consider privacy. However, it has been applied to different application scenarios, where privacy could be a critical issue. Thus, privacy has become a critical issue for its widely successful adoption. Next, we introduce some privacy enhancing techniques to preserve the user anonymity in the blockchain network. Informally, anonymity indicates that the true identity of a user should be concealed from the public while enjoying the blockchain services. In specific, anonymity can be defined in two different levels: pseudonym and unlinkability.

Pseudonym and unlinkability

Pseudonym means that users can choose different pseudonyms for different services. A general method to achieve pseudonym and authenticity at the same time is to use pseudonym-based certificate. This requires a public key infrastructure (PKI) to manage the certificates for users. Instead of pseudonym-based certificates, PKI can also adopt pseudonym-based identity signature for preserving the anonymity of a user. In an identity-based signature, users can derive public keys from the public identities and thus avoid the communication and storage overhead of the certificate.

In pseudonym based methods, user activities can be linked together when they use the same pseudonym across the different services. This is an issue if an attacker has some background knowledge and could launch de-anonymization attacks. Motivated by this issue, another concept is one-time unlinkability which means that users can be anonymously authenticated each time and cannot be linked across different sessions. Existing cryptographic techniques such as group signature and ring signature can achieve this property. In a group

signature scheme, an issuer can assign group credentials to each group member. A group member with valid group certificate can prove the membership in a zero-knowledge manner. Due to the randomness introduced in the proof process, the unlinkability is achieved for different sessions.

Ring Signature

Ring signature is a particular type of digital signature which can provide privacy for signers who want to prove the authenticity of a message to a verifier without revealing his identity. It has been widely utilized in cryptocurrency systems such as Monero to ensure that there is no way for anyone to trace a user only according to his transactions on the blockchain. Specifically, using ring signature technique, Monero can protect the privacy of the sender through obscuring the transaction inputs such that no one can identify the actual signer of the transaction.

Generally, a ring signature is generated by someone in a group and it guarantees that no matter which group member signs on a message, nobody can determine who the group member is. Suppose a group of users each of which has a pair of public/private keys, then any user in the group can generate a ring signature on a message with his private key and all the public keys in the group. The produced ring signature can be publicly verified by anyone with the group of public keys. Taking Monero as an example, suppose that a user A wants to send some Monero coins to user B in order to finish an exchange, the workflow of a ring signature to ensure the privacy of user A who signs the transaction is briefly introduced in the following: First, user A chooses some arbitrary non-signers of the ring signature from past transaction outputs which include their public keys; then user A signs the transaction with his privacy key and public key together with the non-signers' public keys so that it is computationally infeasible for anyone else to identify the actual signer. Note that although user A's public is also used in the signature, it could also be used in other transactions by some others.

2.2 Supply chain system

Digitizing information and its exchange experienced significant improvement in the last few decades by undergoing different stages or phases due primarily to technological innovation and demands. Such changes went from theory-based to the secluded application of several aspects of organizations or their exchange. During this phase, the process was complex with multiple applications and databases, where each one was served by different departments or services such as finance, maintenance, human resources or purchasing. After this phase, the trend was for organizations to go to a new concept, that is integrating the various spectrums of the entity under one system, known as enterprise resource planning (ERP). The ERP system brought a remarkable revolution that merged the digital thread of numerous services within each institution, taking into account the different emphases of each business, depending on the organization's core business. This revolution led to facilitation between all business functions within the entity in addition to managing transactions with their partners. Nevertheless, persistent demand led to globalization of these types of systems, especially for corporate groups, in an attempt to be more broadly linked with their corporate partners. Since then, the digital thread has further extended to include organizations' external relations which, in turn, facilitates and accelerates the wheel of productivity for customer relations management systems "CRMs". Such advances in facilitation in communication processes has led to the emergence of the cloud computing model which allows several entities to exchange data using one common or shared platform. Currently, these frameworks, which are seamlessly incorporated within the internal processes of an organization, play a significant role in improvements related to transparency and efficiency. However, with the evolution of technology, demands become increasingly more accurate.

2.2.1 Digital supply chain (DSC)

The digital supply chain (DSC) has numerous benefits. These include services that are cost effective and activities that create value with numerous benefits to the many players in the ecosystem, including suppliers, firms, customers, and employees. A supply chain is made up of three or more parties: the organization; the flow of products or services, both downstream and upstream, from sources; and consumers [34]. The concept of supply chain reinforces the role played by the flow of information between firms, mainly at activity and business process levels. For efficient integration between players to be realized In order to realize efficient integration between actors, the supply chain must have its processes and information integrated. The main characteristic of the DSC is the strategic and operative information exchange between suppliers which encompasses finance, design, production, research, and/or competition [11]. The information exchange enhances communication between the main players in the chain. Generally, inter-organizational coordination is realized through electronic links that connect information systems, making it possible to automate and digitize the processing of source-to-pay processes in the supply chain that bring together customers and suppliers. The sharing of information and processing functions in the supply chain is not only limited to the business process level. Rather, it also encompasses a considerable volume of data from social media applications, devices and sensors (IoT) [24]. An important component of modern DSCs is the integrated supply chain information models. These are reinforced by the role of service automation and information integration, which act as crucial business drivers. According to Santos and Eisenhardt [23], the main motivation behind the integration of the supply chain is the efficiency that comes with minimal governance , including the cost of exchange with other participants in the ecosystem and individuals within the organization in question. Cost savings realized through information technology make it possible for more information to be frequently and accurately processed from numerous sources around the globe. When information flows are properly automated, the need for manual data entry is eliminated , thus

reducing human error. Even though B2B (Business to Business) [38] integration is widely known to build supply chain efficiency, the low levels of system interoperability that are currently experienced result in high costs of investment despite the fact that the probable benefits have not been achieved. Other advantages DSC include: reduction in the cost of a product or service: enhanced flexibility in the supply chain design: reduced lead times in the supply chain: and the creation of a competitive advantage. For information to be considered as effectively shared, it must bring new value to customers and actors in the supply chain in terms of services, visibility, prediction, and decision making. The driving factor is the need to communicate the right information to the right audience at the right time for the purpose of making decisions. The important role of service integration coupled with information integration act as crucial drivers of the value of a business in supply chains. The bundling of information coupled with systemic integration are additional value drivers that create extra value for customers. Currently, attention is being paid to examining value creation from big data in supply network environments that embrace industrial B2B. The focus is also on the inter-organizational integration founded on blockchain technology in the present day economy. The running of supply chains and similar business models may be broadened significantly through innovative information exchange services. For instance, according to Kagermann et al [22], an industry environment is made up of manufacturing systems that are vertically connected to business networks within businesses and factories. Additionally, the processes are connected horizontally to value networks that are disperse in nature manageable in real time. New systemic value elements are introduced through the solid integration of information for public and industrial service users as well as service providers. The focus of transactional supply chain business processes is the development of digital ecosystems which results in numerous business opportunities for the players in the ecosystem. Previous research on the integration of the systemic global supply chain highlighted four requirements for digital business ecosystems. These are the basis for developing business as well as innovation and the present research.

The purpose of the present research is to solve the problem of having a fair goods exchange under a trusted party between merchant and supplier that may suffer from the weakness of single point failure. There exist repudiation attacks that lead to unfair solution evaluation. With the advancement of blockchain technology, research has been conducted to ensure fair data exchange without any third party. Buyers can share their data only after they receive rewards while sellers pay only after they obtain the data. Researchers have commonly adopted the Hashed Timelock contract to make users less likely to behave maliciously. Further research studies have been conducted on fair computation among multiple users [4]. For many years, fair exchange has been extensively studied in order to find a solution to this problem. At the same time, it has been proved or otherwise assumed that fairness in exchange can be achieved without having a Trusted Third Party (TTP) [17, 30, 45]. Others tried to circumvent this impossibility by studying weaknesses on the security models, where the use of a trusted third party can be eliminated only in the case of the failure of either party to abide by the prior commitment agreed upon or if departing from the expected behaviour [10]. Some researchers have studied how the blockchain, or specifically the smart contract, will be a solution where it can take the role of the TTP [25, 33]. The cut-and-choose protocol [10], which is based on investigating the garbled circuits that are exchanged between two parties, is able to expose cheater or misbehaving party. However, it will incur much communication overhead for a blockchain network, which is a p2p network. Setting a smart contract based on that protocol will be high in terms of Gas cost [27]. From another perspective, protocol in [15] has been proposed to achieve fair exchange of digital goods. It has achieved a paradigm shift in a negligible error rate and small costs.

Chapter 3

Secure and Fair IIoT-Enabled Supply Chain Management via Blockchain-based Smart Contracts

3.1 Problem formulation

The focus of this study is a system that enables flexible and efficient on-demand exchange of goods between merchants and suppliers and that specifically addresses the lack of mutual trust between entities, which is one of the major challenges in blockchain technology. In order to respond to this challenge, this present work proposes a fair and transparent supply chain system. This chapter presents the system model, followed by a discussion of the Threat model and design goals.

3.1.1 System model

In the proposed system model, three main entities are involved, as shown in Fig. 3.1. Each entity complements the other and has a specific, clear role. These entities and their roles are described as follows:

- **The Merchant:** Uniquely identified by M, the merchant requires specific goods, and sends an order to the supplier through the blockchain.
- **The Supplier:** Uniquely identified by S, the supplier posts the goods on the blockchain, receives the order from M, and then delivers the goods to M.
- **Blockchain:** The blockchain is the underlying decentralized P2P network. It runs smart contracts and acts as an external judge between the two parties to effectively complete the exchange of goods.

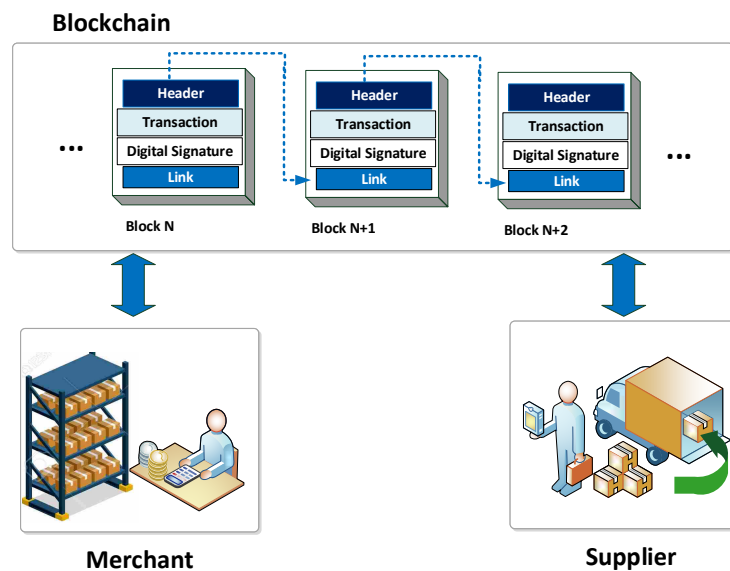


Figure 3.1: System Model

In supply chain management, the most important concept is that of fairness. If someone provides a service, the other party must accommodate by responding, and vice versa. At a high level, the process of fair goods exchange in this proposed system works as follows: Supplier S first sets an item list that designates the goods that Merchant M can order. M then places the order. Subsequently, S processes the order and asks for confirmation from M, who will check the goods information and send an acceptance notification to the smart contract. In the next step, the smart contract will create a tag used for labelling the goods, and publish it so that S can attach it to the goods. S then delivers the goods to M, who

scans the tag after checking the goods, and then submits it to the smart contract. The smart contract in blockchain takes the role of an external judge based on the information collected to determine whether if the contract is correctly fulfilled or to enforce penalties to the misbehaving parties.

3.1.2 Threat model

The goal of the proposed system is to realize a fair goods exchange between the two entities, Supplier and Merchant, who could be malicious or untrusted. In particular, the threat model can be summarized as follows:

- **Security against selfish/malicious suppliers:** The suppliers could be selfish or malicious users who may not send the goods on time. Moreover, a malicious supplier can also send different goods other than those agreed in the contract.
- **Security against selfish/malicious merchants:** The merchants could also be selfish or malicious. They may not submit the scan or send the signature to the smart contract after receiving the goods. It is also possible for merchants to reject the goods and argue that they are not what was requested, which will cost the supplier to lose the deposit on the smart contract.

3.1.3 Design goals

The design goals mainly contain three aspects:

- **Fairness:** Our protocol guarantees fairness by relying on smart contracts between suppliers and merchants over decentralized cryptocurrency. An honest supplier must ensure that the merchant will pay after receiving the goods and an honest merchant is assured that he only pays if the supplier delivers the goods ordered by the merchant. In cases of disagreement, in the proposed system, the contract takes the role of a

judge. In other words, the smart contract goes beyond a traditional contract that represents an agreement between the supplier and merchant, and also acts as a judge to resolve disputes between the supplier and merchant.

- **Transparency:** One of the benefits provided by the proposed protocol is that it enhances the demand for transparency, ensuring that the transaction process is transparent and immutable to both parties.
- **Proof-of-concept implementation:** Implementation of a fair goods exchange smart contract between two nodes is presented and explained in detail in Chapter 4, which also demonstrates the functionality of the proposed scheme. In addition, results in terms of the cost of gas that will be paid to process each phase in our smart contract in the blockchain, as well as transaction confirmation time, are analyzed and simulated.

3.2 Proposed Scheme

3.2.1 The System architecture

This subsection provides an overview of the system architecture considered in this work. In completing an exchange, or in cases of disagreement, the smart contract take the role of an external judge between two entities. It is assumed that Supplier publishes a stock information of the goods into the blockchain. Thus, both parties start by registering themselves as smart nodes. As shown in Fig. 3.2 , Merchant M will place the order that contains all the information about the goods he wants, in addition to certain conditions such as the delivery deadline, which will be addressed in detail in Section 3.2.2.4. S will then respond with approval after confirming the availability of the goods through the blockchain; this is considered as the third stage in the process.

The smart contract will generate the tags which contain the goods information and the deadline for the delivery, as previously agreed upon by the parties. The tags will be

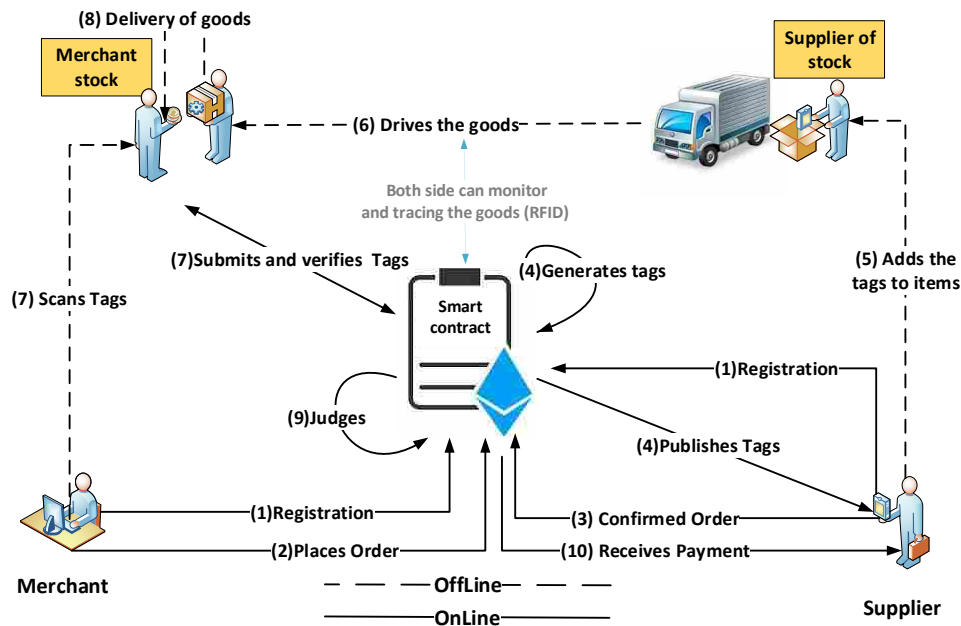


Figure 3.2: System Architecture

available for both entities on the smart contract. Thereafter, Suppliers will print the tags and add them to the goods. The goods are sent in the sixth stage with the use of RFID [19, 47], which will enable both parties to track shipments of the goods from the smart contract. When the goods arrive at Merchant’s place, M will scan the tags on the goods after inspecting them. This indicates that M confirms acceptance on the goods after confirmation of the smart contract of the validity of the tags and the delivery time. Finally, M is given the goods and signs for them. The smart contract will then be notified that the goods have been successfully delivered, and Suppliers will retrieve the deposit in the smart contract.

3.2.2 The Scheme

This section provides the details of the smart contract construction, which assumes that secure channels [43, 44] have been established between all entities. For example, SSL (Secure Sockets Layer) can be used to establish a secure channel between two entities [32]. In the proposed scheme, as shown in Fig. 3.3, there are five phases, each of which is based on the previous phase, with the exception of the first phase. All notations are explained in

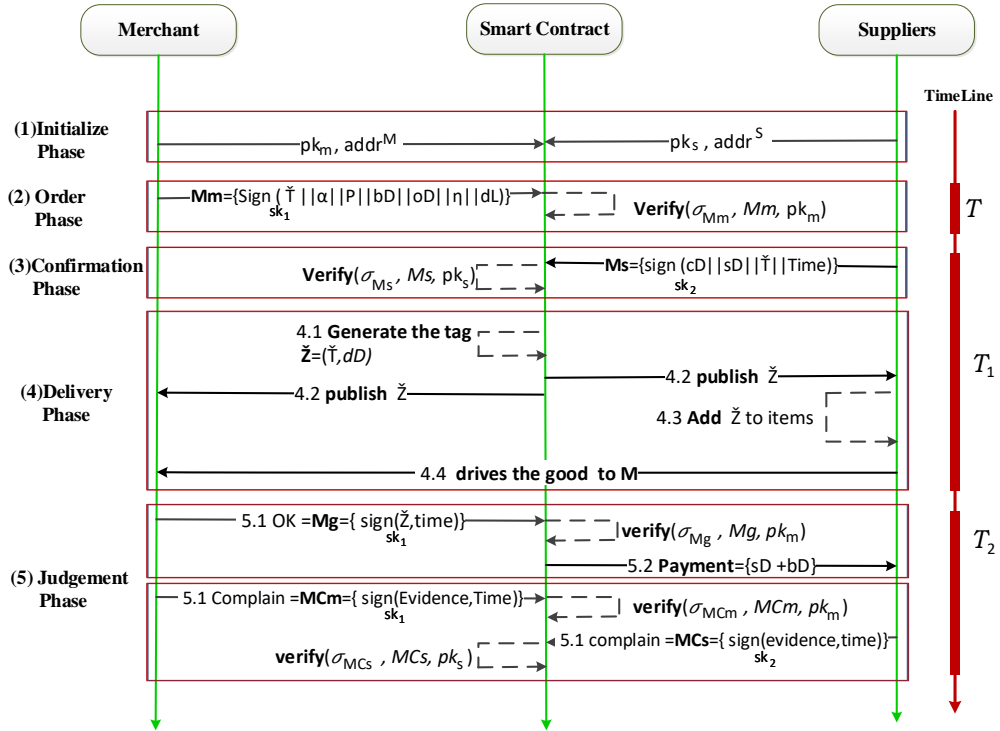


Figure 3.3: Outline of fair goods exchange

Table 3.1.

3.2.2.1 Initialize Phase

Both M and S will register their addresses and public keys to the smart contract as shown in the algorithm 1. More information can be added, including $pk_m, addr^M$, where pk_m is the public key and $addr^M$ is M's address on the smart contract. For his part, S will register $pk_s, addr^S$ where pk_s is the public key and $addr^S$ is S's address. In the blockchain, every account is described by a pair of keys, namely a public/private ECDSA keypair. User address is obtained from the last 20 bytes of their public key, so it become a 160-bit of hash. Using private-key cryptography is to "sign" data then the other side will verify that the signature is valid by using the public key of the sender to achieve integrity of the data.

Notation	Explanation
$addr^M$	The Merchant's address
$addr^S$	The Supplier's address
$Sign_k^m$	Signature of Message m signed by k. Messages include M_m =Order Phase, M_s =Confirmation Phase, M_g =Judgement OK, M_{C_m} =Judgement complain M, M_{C_s} =Judgement complain S)
pk_m	Merchant's public key
pk_s	Supplier's public key
sk_1	Merchant's private key
sk_2	Supplier's private key
\hat{T}	Item
α	A variable between $0 \leq \alpha \leq 1$, that determines the ratio of P that each of M and S will deposit
P	The price
bD	Merchant's deposit
oD	The order date
η	The number of \hat{T}
σ_m	Signed message
dL	Delivery deadline
v, r, s	r and s are outputs of an ECDSA signature, and v is the recovery id
cD	Confirmed date
sD	Supplier's deposit
\hat{Z}	Tags
dD	Delivery Date
T_1	Time refers to interval between confirmation and delivery
T_2	Time refers to the time limit for participators including, M and S to upload evidence (proof of misbehavior)

Table 3.1: Notation explanation

3.2.2.2 Order Phase

M will place the order M_m and send it to S after signing the message with his secret key sk_1 for authentication where it will be verified by miners on the blockchain by his pk_m . The format of the order M_m is shown below:

$$M_m = (\hat{T} \parallel \alpha \parallel P \parallel bD \parallel oD \parallel \eta \parallel dL)$$

where \hat{T} represents the low stock threshold for an item, which is used to determine whether an order should be placed according to the current stock of the item, i.e., lower

Algorithm 1: Initialize Phase

```
1 function Initialization  $pk_m, addr^M pk_s, addr^S$ ;  
2     M public key = M public key;  
3     S public key = S public key;  
4     M Address = M Address;  
5     S Address = S Address;
```

than \hat{T} , α represents a variable ranging from 0 to 1 ($0 \leq \alpha \leq 1$) and determines the ratio of price P that M and S will deposit, oD refers to the order date, η includes the amount of goods required by M , and dL represents the delivery deadline. M deposits bD after calculating the αp with η . It should be noted that α is used to ensure that M will honor the order and prevent M from dishonestly cancelling the order.

Algorithm 2: Order Phase

```
1 function OrderPhase ( $\hat{T}, \eta, \sigma_{M_m}, dL, addr^M, \alpha, P, v, r, s, depositAccount$ );  
2      $\triangleright$  check the identification of Merchant;  
3     require( $msg.sender == addr^M$ )  
4      $\triangleright$  check the signature of Merchant;  
5     require( $verifySignature(addr^M, \sigma_{M_m}, v, r, s)$ )  
6      $\triangleright$  add new order;  
7     newOrder  $\leftarrow \{\hat{T}, now, \eta, dL\}$   
8     memory orderList[i++]  $\leftarrow$  newOrder  
9      $\triangleright$  calculate the price ;  
10    Total P  $\leftarrow \eta * \hat{T}.P$   
11     $\triangleright$  check the balance of the account;  
12    require( $msg.value \geq totalPrice * (p + \alpha/100)$ )  
13     $\triangleright$  Merchant Deposit ;  
14    depositAccount += amt  
15     $\triangleright$  Phase status ;  
16    newOrder.status  $\leftarrow$  Initialized
```

Then, M will send the signed order to S after checking his \hat{T} threshold. The smart contract checks the identification of Merchant $addr^M$ and then adds a new order in the order list which contains \hat{T} , oD , η , P and dL as input. As shown in the algorithm 2, the smart contract will calculate the total price of the goods by $P * \eta$. M will send the bD to the smart contract address. The entire message M_m will be signed by sk_1 using (v, r, s) ,

where r and s are outputs of an *ECDSA* signature, and v is the recovery id [21], and the resulting signature is $Sign_{sk_1}^{M_m}$.

3.2.2.3 Confirmation Phase

In this phase, S will confirm the order based on the availability of the product that he has already been presented on the public blockchain, including the cD which is the confirmation day, the \hat{T} which is the item that he will finally deliver. For authentication purposes, S will sign the message with his secret key sk_2 , where it will be verified by miners on the blockchain by his pk_s .

Algorithm 3: Confirmation Phase

```

1 function ConfirmationPhase ( $\hat{T}, i, \sigma_{M_s}, addr^S, cD, v, r, s, \eta, \alpha, depositAccount$ );
2      $\triangleright$  check the identification of Supplier ;
3     require( $msg.sender == addr^S$ )
4      $\triangleright$  check the signature of Supplier;
5     require( $verifySignature(addr^S, \sigma_{M_s}, v, r, s)$ )
6      $\triangleright$  Read the latest item from the order list ;
7     memory latestOrder  $\leftarrow$  orderList[i]
8     latestOrder.cD  $\leftarrow$  now
9      $\triangleright$  check the deposit;
10    totalDeposit  $\leftarrow \eta * p * (\alpha/100)$ 
11    require( $msg.value > totalDeposit$ )
12     $\triangleright$  Supplier Deposit ;
13    depositAccount += amt
14     $\triangleright$  Phase status ;
15    newOrder.status  $\leftarrow$  Confirmed

```

The second phase is called the confirmation phase, where S will confirm the order, then M can start delivering the goods. This phase begins with checking the identification of S $addr^S$ and then checking if M has given the deposit based on $P * \eta$ and α of the deposit to prevent unexpected and dishonest cancellation of the order. M_s will then be signed for the signature $Sign_{sk_2}^{m_s}$ by sk_2 . At that point, the status of the contract will be *confirmed*, as is shown in Algorithm 3.

3.2.2.4 Delivery Phase

This phase contains offline and online processes that start with generating the \check{Z} on the smart contract for goods that S will deliver. The \check{Z} will contain \hat{T} and dD , which is the delivery date. The smart contract will generate \check{Z} after S confirmed the order in the confirmation phase as is shown in Algorithm 4. In the final stage, tags will then be published to both parties and will be available through the smart contract. Afterwards, S processes the goods for delivery after labelling them. In the delivery phase, the smart contract will generate \check{Z} based on the information that is given in the order and confirmation phases $\check{Z}=(\hat{T},dD)$. Having been generated, the tags will be available for both sides. At this point, S will use these tags, add them to the goods and deliver the goods to M.

Algorithm 4: Delivery

```

1 function GenerateTags;
   Input  :  $\hat{T},dD$ 
   Output:  $\check{Z}$ 
2     Order memory  $\leftarrow$  orderList[index];
3      $\triangleright$  Generate tag
       tagsA  $\leftarrow$  keccak256 (abi.encodePacked( $\hat{T},dD$ ));
4     return tagsA ;

```

3.2.2.5 Judgement Phase

This phase has two paths to either satisfy both parties, which means (ok) as shown in Algorithm 5, or not satisfy, which means (complaint), as shown in Algorithm 6. In a satisfying path as presented in Algorithm 5, M will scan the tags \check{Z} , and miners will verify the $Sign_{sk_1}^{M_g}$ by using pk_m . After the scan, the smart contract will check the time of the delivery by checking the deadline that should be agreed by both sides in the previous phase according to (dD, T_1, T_2) . If condition $now > (dD + T_1 + T_2)$ is met. S will then withdraw $\eta * P * (1 + 2 \cdot \alpha/100)$, which means that S will receive the two deposits and the status phase becomes *Complete*.

Algorithm 5: Judgement (OK)

```
1 function AcceptOrder( $T_1$ ,  $addr^M$ ,  $cD$ ,  $\sigma_{M_g}$ ,  $v$ ,  $r$ ,  $s$ , depositAccount);
2     ▷ check the identification of Merchant
   require( $msg.sender == addr^M$ );
3     ▷ check the signature of Merchant
   require( $verifySignature(addr^M, \sigma_{M_g}, v, r, s)$ );
4     ▷ check the balance of the account
   require( $(now - cD) < T_1$ );
5 function OrderPayment ( $dD$ ,  $T_1$ ,  $T_2$ ,  $addr^M$ ,  $addr^S$ ,  $\eta$ ,  $P$ ,  $\alpha$ );
6     require( $now > dD + T_1 + T_2$ );
7     require( $msg.sender == addr^M$ );
8     ▷ withdraw(unit256 amount, unit256 depositAccount, address receiver):
   withdraw the deposit amount from the contract
   account depositAccount to the address receiver;
9     withdraw( $\eta * P * (\alpha / 100)$ ), depositAccount,  $addr^M$ );
10    require( $msg.sender == addr^S$ );
11    withdraw( $\eta * P * (1 + \alpha / 100)$ ), depositAccount,  $addr^S$ );
12    ▷ Phase status
   newOrder.status ← Completed;
```

Algorithm 6: Judgement (Complaint)

```
1 function proofOfComplaint( $addr^M$ ,  $addr^S$ , evidence,  $dD$ ,  $T_1$ ,  $T_2$ ,  $M_{C_s}$ ,  $\alpha$ ,  $P$ ,  $M_{C_m}$ ,
    $v$ ,  $r$ ,  $s$ );
2     require( $now > dD + T_1$  AND  $now < dD + T_1 + T_2$ );
3     require( $verifySignature(msg.sender \sigma_{M_{C_s}}, v, r, s)$ );
4     require( $verifySignature(msg.sender \sigma_{M_{C_m}}, v, r, s)$ );
5     require( $msg.sender == addr^M$ );
6     ▷ If the evidence comes from merchant, he can take both the deposit
   withdraw( $\eta * P * (1 + 2 * \alpha / 100)$ ), depositAccount,  $addr^M$ );
7     require( $msg.sender == addr^S$ );
8     ▷ If the evidence comes from supplier, he can tak both the deposit
   withdraw( $\eta * P * (2 * \alpha / 100)$ ), depositAccount,  $addr^S$ );
9     return;
```

As previously mentioned, T_2 is the period specified prior to the expiration of the agreement which allows either party to provide proof of complaint as shown in Algorithm 6. The function *proofOfComplaint* will first check the timeline which requires that $now > (dD + T_1)$ and $now < (dD + T_1 + T_2)$. Both sides can provide evidence, e.g., pictures or tracing records of the goods. In the case that one side provides clear evidence before the time limit is expired, judiciary is made in the smart contract based on the provided evidence. Particularly, if M provides the evidence to complain that S has the inappropriate behaviors, then M can take both deposits and the money of goods back. Otherwise, S can take both deposits by providing valid evidence. In addition, the case of neither party objecting to any initiative or confirmation of the receipt within the timeline, the amount will be transferred from the smart contract account and returned separately to both parties.

3.3 Security analysis

This subsection provides an analysis of the security of the proposed scheme by providing an intuition on why neither party can break the fairness property. As mentioned in the previous section, the Judgement phase is the payout phase which is the last round where both entities can be satisfied. The contract is terminated to enable the supplier to receive a full deposit or punish the misbehaving parties. If no one can provide a valid evidence before T_2 , then the deposit will be sent back to each party. This is the point at which the smart contract acts as the role of judgement in the case of disagreement to complete the contract in a fair way. The transaction processes and status in all phases are transparent and immutable for both parties. As a result, traceability in the proposed exchange protocol is also achieved through the provision of the smart contract that allows both entities to trace the transactions and contract states that are reached.

Chapter 4

Experiment Results

4.1 Experiment setup framework

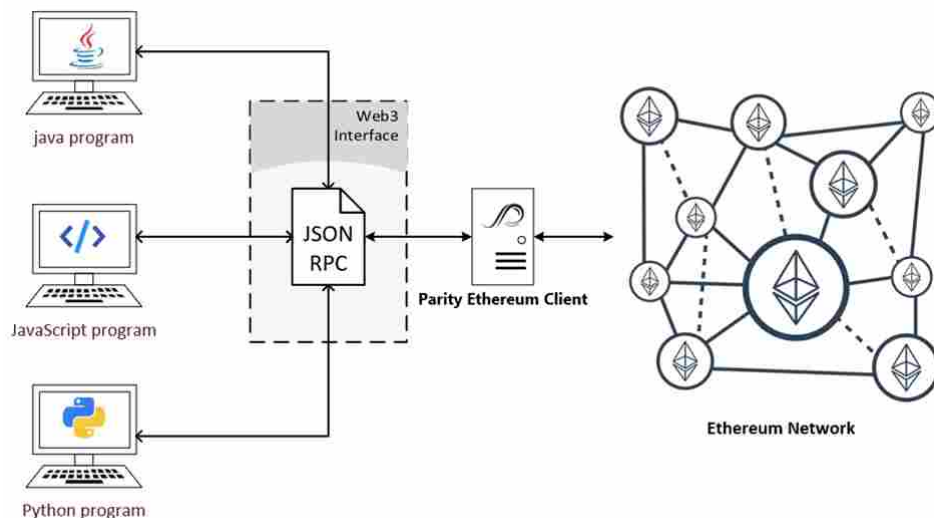


Figure 4.1: Structure of the Parity Ethereum Client

As proof of concept, it is important to implement a working prototype of the proposed system. Considering that it is the lightest and fastest Ethereum client, using Parity as the test platform is a good choice [14]. This platform, written in a programming language called Rust, provides improved performance, code clarity and reliability. Parity Ethereum Client v2.0 comes with options of building on different operating systems. The JSON-

RPC HTTP and Web-Sockets server on this platform run by default on port 8545 and 8546. Parity, which is an alternative to Gath, the official Ethereum Client, has many more features and is more quickly able to build the decentralized application that is needed to connect to the Ethereum blockchain. In addition to supporting several APIs,* it is fully configurable [14]. As shown in Fig. 4.1, the client connects to the blockchain by using the web3 interface. The simulate of the proposed system was implemented on a laptop with 2.40 GHz Intel Core i5 processors and 8 GB memory on Ubuntu Linux Operating System. The next section is the steps of building Parity Ethereum Client.

4.1.1 Building a Parity Ethereum Client

1. After login as root, use the following command to download the Parity binary from the server.

```
bash <(curl https://get.parity.io -L)
```

2. Build the essential tools for Ubuntu. These are mainly compilers and libraries that will help to build the *Web3* client and *node.js*. Update the package manager.

```
apt-get update
```

3. Build essentials by following commands that will ask a question about the performance. Give permission to continue.

```
apt-get install build-essential
```

4. Install *Node.js* from the official web site:

https : //github.com/nodesource/distributions/blob/master/README.md

Now download the package from *Alternatively, for Node.js10*

```
$curl -sL https://deb.nodesource.com/setup_10.x
```

```
$sudo -E bash -$
```

5. Update everything once again, then access all the library needed for *Node.js*. Install *Node.js*.
-

```
$sudo apt-get
$sudo install -y nodejs$
```

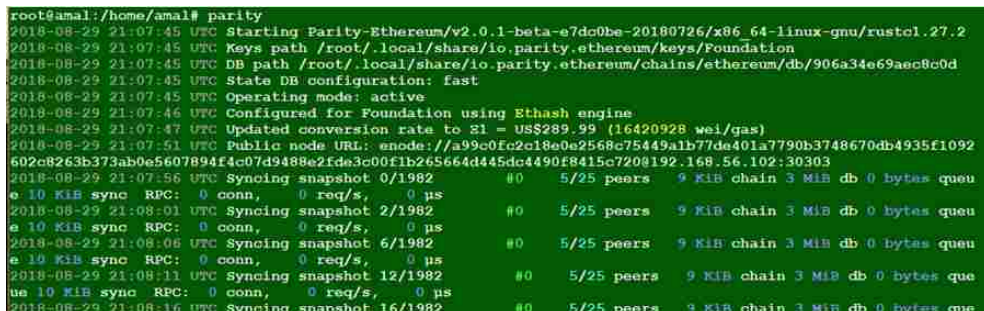
6. Install the Web3 JavaScript library by first installing npm, then web3, in order to connect to the Parity client.
-

```
npm install web3
```

When all the installation processes are positively finished, the technical tests can start by following the next section.

4.1.1.1 Technical Tests

1. In order to check the connection of the Parity client with the official public parity network, *parity* runs in the terminal. As shown in Fig. 4.2, this starts by syncing with the main Ethereum official public blockchain.



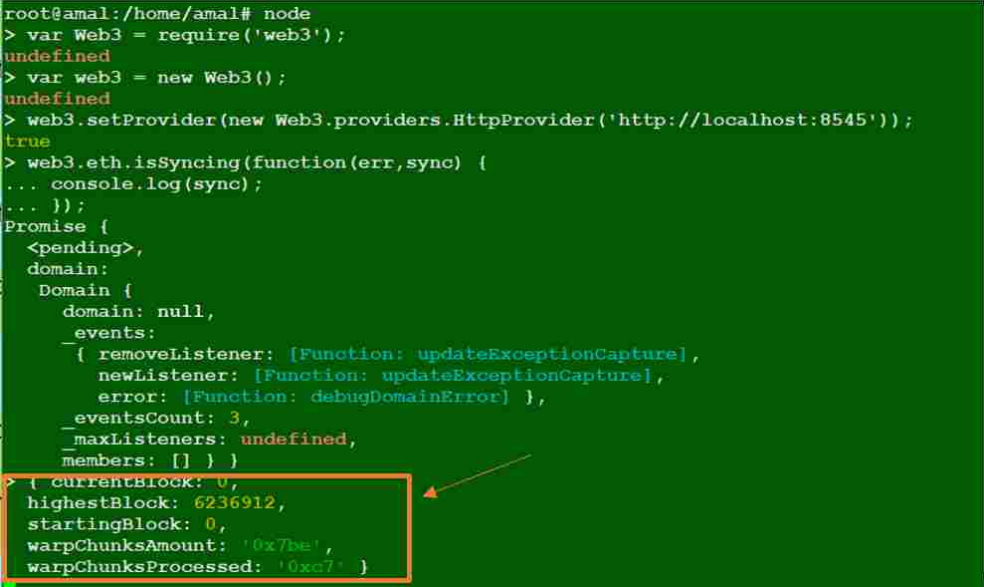
```
root@amal:/home/amal# parity
2018-08-29 21:07:45 UTC Starting Parity-Ethereum/v2.0.1-beta-e7dc0be-20180726/x86_64-linux-gnu/rustcl.27.2
2018-08-29 21:07:45 UTC Keys path /root/.local/share/io.parity.ethereum/keys/Foundation
2018-08-29 21:07:45 UTC DB path /root/.local/share/io.parity.ethereum/chains/ethereum/db/906a34e69aec8e0d
2018-08-29 21:07:45 UTC State DB configuration: fast
2018-08-29 21:07:45 UTC Operating mode: active
2018-08-29 21:07:46 UTC Configured for Foundation using Ethash engine
2018-08-29 21:07:47 UTC Updated conversion rate to 31 = US$289.99 (16420928 wei/gas)
2018-08-29 21:07:51 UTC Public node URL: enode://a99c0fc2c18e0e2568c75449a1b77de401a7790b3748670db4935f1092
602c8263b373ab0e5607894f4c07d9488e2fde3c00f1b265664d445dc4490f8415c7208192.168.56.102:30303
2018-08-29 21:07:56 UTC Syncing snapshot 0/1982 #0 5/25 peers 9 KiB chain 3 MiB db 0 bytes queu
e 10 KiB sync RPC: 0 conn, 0 req/s, 0 μs
2018-08-29 21:08:01 UTC Syncing snapshot 2/1982 #0 5/25 peers 9 KiB chain 3 MiB db 0 bytes queu
e 10 KiB sync RPC: 0 conn, 0 req/s, 0 μs
2018-08-29 21:08:06 UTC Syncing snapshot 6/1982 #0 5/25 peers 9 KiB chain 3 MiB db 0 bytes queu
e 10 KiB sync RPC: 0 conn, 0 req/s, 0 μs
2018-08-29 21:08:11 UTC Syncing snapshot 12/1982 #0 5/25 peers 9 KiB chain 3 MiB db 0 bytes que
ue 10 MiB sync RPC: 0 conn, 0 req/s, 0 μs
2018-08-29 21:08:16 UTC Syncing snapshot 16/1982 #0 5/25 peers 9 KiB chain 3 MiB db 0 bytes que
```

Figure 4.2: Connecting with the official public blockchain

2. Connect the local Parity with web3, as shown in Fig. 4.3, which opens a new terminal

while keeping the Parity public blockchain terminal running. This is followed by login as root and running *Node.js* by the *node*, then issuing the following command:

```
> var Web3 = require('web3'); // load up web3
undefined
> var web3 = new Web3(); // To create new client instance
undefined
> web3.setProvider(new
    Web3.providers.HttpProvider('http://localhost:8545')); //
    To connect to the localhost parity client
true
> web3.eth.isSyncing(function(err, sync) {
... console.log(sync);
... }); //Syncing will provide data that shows that:
```



```
root@amal:/home/amal# node
> var Web3 = require('web3');
undefined
> var web3 = new Web3();
undefined
> web3.setProvider(new Web3.providers.HttpProvider('http://localhost:8545'));
true
> web3.eth.isSyncing(function(err, sync) {
... console.log(sync);
... });
Promise {
  <pending>,
  domain:
    Domain {
      domain: null,
      _events:
        { removeListener: [Function: updateExceptionCapture],
          newListener: [Function: updateExceptionCapture],
          error: [Function: debugDomainError] },
      _eventsCount: 3,
      _maxListeners: undefined,
      members: [] }
}
{ currentBlock: 0,
  highestBlock: 6236912,
  startingBlock: 0,
  warpChunksAmount: '0x7be',
  warpChunksProcessed: '0xc7' }
```

Figure 4.3: Connected with web3

It can be observed that, on the main Ethereum blockchain, if they are not connected, the blocks highlighted in Fig 4.3 will show up as undefined.

4.1.2 Building Proof-of-Authority Chains

Proof of Authority (PoA) is supported by Parity with consensus power to be used with Ethereum Virtual Machine based chains. PoA is a new type of consensus function on blockchain that can be useful for a private chain. PoA does not solve an arbitrarily difficult algorithm that depends on random nodes, but instead uses a group of authority nodes that are apparently able to solve the algorithm, then creates a new secure block in the blockchain. The chain in PoA must be signed off and approved by most of the authorities to become a part of the permanent record. This makes it easy to check and maintain the private chain that keeps the block issuers accountable. In this experiment, multiple accounts are created; one is a user account, and two are authorities. Each authority account will usually only run in a single mode. However, all accounts here are run in the same node.

1. Starting with configuration, create the config file (*config.toml*) with the following fields and post it in path */.local/share/io.parity.ethereum/config.toml*

```
[parity]
chain = "/root/demo-spec.json " //the path of json file in
      the computer
[account]
pwds = ["/root/node.pwds "] ////the path of the node
      passwords file in your computer
[mining]
engine_signer = "0x37f93cfe411fa244b87ff257085ee360fca245e8"
reseal_on_txs = "none" // automatic authorities reseal
```

2. Create a *demo – spec.json*, which contains the basic chain required fields as shown in Fig. 4.4, and save it under *demo – spec.json* in */root/* or the same path that added it to the config file.
3. Start to set up the two nodes. Before setup, it must be understood that Parity uses


```

1 [parity]
2 chain = "demo-spec.json"
3 base_path = "/tmp/parity0"
4 [network]
5 port = 30300
6 [rpc]
7 port = 8540
8 apis = ["web3", "eth", "net", "personal", "parity", "parity_set", "traces", "rpc", "parity_accounts"]
9 [ui]
10 port = 8180
11 [dapps]
12 port = 8080
13 [account]
14 password = ["node.pwds"]
15 [mining]
16 engine_signer = ""
17 reseal_on_txs = "none"

```

Figure 4.6: configure Node 0

```

1 [parity]
2 chain = "demo-spec.json"
3 base_path = "/tmp/parity1"
4 [network]
5 port = 30301
6 [rpc]
7 port = 8541
8 apis = ["web3", "eth", "net", "personal", "parity", "parity_set", "traces", "rpc", "parity_accounts"]
9 [ui]
10 port = 8181
11 [dapps]
12 port = 8081
13 [account]
14 password = ["node.pwds"]
15 [mining]
16 engine_signer = ""
17 reseal_on_txs = "none"
18 [websockets]
19 port = 8547

```

Figure 4.7: configure Node 1

- Run the node 0 by `parity --config node0.toml` and it will return an address.

Authority address 1

```

curl data '{
    "jsonrpc": "2.0",
    "method": "parity_newAccountFromPhrase",
    "params": ["node0", "node0"],
    "id": 0
}' -H "Content-Type: application/json" -X
    POST localhost:8540

```

```

root@amal:~# curl --data '{"jsonrpc":"2.0","method":"parity_newAccountFromPhrase","params":["node0", "node0"],"id":0}' -H "Content-Type: application/json" -X POST localhost:8540
{"jsonrpc":"2.0","result":"0x00bd138abd70e2f00903269f3db08f2d25677c9e","id":0}

```

Figure 4.8: Authority address 1

User address

```
curl --data '{
    "jsonrpc": "2.0",
    "method": "parity_newAccountFromPhrase",
    "params": ["user", "user"],
    "id": 0
}' -H "Content-Type: application/json" -X
    POST localhost:8540
```




```
root@amal:~# curl --data '{"jsonrpc": "2.0", "method": "parity_newAccountFromPhrase", "params": ["user", "user"],
"jsonrpc": "2.0", "result": "0x004ac07d2329597267ac62b4166639513386f32a", "id": 0}' -H "Content-Type: application/json" -X POST localhost:8540
```

Figure 4.9: User address

- Run the node 1 by parity `-config node1.toml` and it will return an address.

Authority address 2

```
curl --data '{
    "jsonrpc": "2.0",
    "method": "parity_newAccountFromPhrase",
    "params": ["node1", "node1"],
    "id": 0
}' -H "Content-Type: application/json" -X POST
    localhost:8541
```



```
root@amal:~# curl --data '{"jsonrpc": "2.0", "method": "parity_newAccountFromPhrase", "params": ["node1", "node1"],
"jsonrpc": "2.0", "result": "0x00aa39d30f0d20ff03a22ccfc30b7efbfa597c2", "id": 0}' -H "Content-Type: application/json" -X POST localhost:8541
```

Figure 4.10: Authority address 2


```

1 [parity]
2 chain = "demo-spec.json"
3 base_path = "/tmp/parity1"
4 [network]
5 port = 30301
6 [rpc]
7 port = 8541
8 apis = ["web3", "eth", "net", "personal", "parity", "parity_set", "traces", "rpc", "parity_accounts"]
9 [ui]
10 port = 8181
11 [dapps]
12 port = 8081
13 [account]
14 password = ["node.pwds"]
15 [mining]
16 engine_signer = "0x00Aa39d30F0D20FF03a22cCfc30B7EfbFca597C2"
17 rescall_on_txs = "none"
18 [websockets]
19 port = 8547

```

Figure 4.13: Completed file Node 1

```

root@amal:~# parity --config node0.toml
Loading config file from node0.toml
Option '--dapps-port' has been removed and is no longer supported.
2018-09-01 21:18:24 UTC Starting Parity-Ethereum/v2.0.1-beta-e7dc0be-20180726/x8
6_64-linux-gnu/rustcl.27.2
2018-09-01 21:18:24 UTC Keys path /tmp/parity0/keys/DemoPoA
2018-09-01 21:18:24 UTC DB path /tmp/parity0/chains/DemoPoA/db/d0678730db7ea493
2018-09-01 21:18:24 UTC State DB configuration: fast
2018-09-01 21:18:24 UTC Operating mode: active
2018-09-01 21:18:25 UTC Configured for DemoPoA using AuthorityRound engine
2018-09-01 21:18:31 UTC Public node URL: enode://a5907b6f6a36336cee6cda82df148b8
907ae73a310dc6b5e603bf2082f9dlce5d02a96f6debcf7039469e4fecl3b545130fcd0c73871481
f28cf0ae46b8c43d8@192.168.56.102:30300
2018-09-01 21:19:00 UTC 0/25 peers 9 KiB chain 7 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 0 µs
2018-09-01 21:19:30 UTC 0/25 peers 9 KiB chain 7 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 289019 µs
2018-09-01 21:20:00 UTC 0/25 peers 9 KiB chain 7 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 289019 µs
2018-09-01 21:20:30 UTC Imported #1 0x6135..920c (0 txs, 0.00 Mgas, 300 ms, 0.57
KiB)
2018-09-01 21:20:30 UTC 0/25 peers 9 KiB chain 8 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 289019 µs
2018-09-01 21:21:00 UTC 0/25 peers 9 KiB chain 8 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 289019 µs
2018-09-01 21:21:30 UTC 0/25 peers 9 KiB chain 8 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 289019 µs
2018-09-01 21:22:00 UTC 0/25 peers 9 KiB chain 8 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 289019 µs
2018-09-01 21:22:30 UTC 0/25 peers 9 KiB chain 8 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 289019 µs
2018-09-01 21:22:40 UTC Imported #2 0xd804..116f (0 txs, 0.00 Mgas, 0 ms, 0.57 Ki
B)

```

Figure 4.14: Run node0

```

root@amal:~# parity --config node1.toml
Loading config file from node1.toml
Option '--dapps-port' has been removed and is no longer supported.
2018-09-01 21:18:39 UTC Starting Parity-Ethereum/v2.0.1-beta-e7dc0be-20180726/x8
6_64-linux-gnu/rustcl.27.2
2018-09-01 21:18:39 UTC Keys path /tmp/parity1/keys/DemoPoA
2018-09-01 21:18:39 UTC DB path /tmp/parity1/chains/DemoPoA/db/d0678730db7ea493
2018-09-01 21:18:39 UTC State DB configuration: fast
2018-09-01 21:18:39 UTC Operating mode: active
2018-09-01 21:18:40 UTC Configured for DemoPoA using AuthorityRound engine
2018-09-01 21:18:45 UTC Public node URL: enode://b3abef287c4140942a1dd27c98d17f0
927a9879bbaa8a7d465a1f9af0bea091c6c8f0471763adf016851b0194d6alcf682eea9a3af6e8f9
56e8b3d14e93b915c@192.168.56.102:30301
2018-09-01 21:19:10 UTC 0/25 peers 9 KiB chain 7 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 0 µs
2018-09-01 21:19:40 UTC 0/25 peers 9 KiB chain 7 KiB db 0 bytes queue 448 b
ytes sync RPC: 0 conn, 0 req/s, 0 µs
2018-09-01 21:20:10 UTC 0/25 peers 9 KiB chain 7 KiB db 0 bytes queue 448 b

```

Figure 4.15: Run node1

4.1.2.1 Technical Tests

1. Connect the nodes by using curl, then keep the two terminal nodes open and use a new terminal with this command:

```
curl --data '{
    "jsonrpc": "2.0",
    "method": "parity_enode",
    "params": [], "id": 0
}' -H "Content-Type: application/json" -X POST
localhost:8540
```

The result is enode://24829e7ac2505b63256dbbfcc a5e041d84834e9d30c083869fb20da389b1e3d36d5096a503b9296b785eba5c2a57dc 6af81b8bbb3d601cb72dbe7624e8ea7347@192.168.56.103:30300 . Now add the result from node 0 to node 1 in the following command:

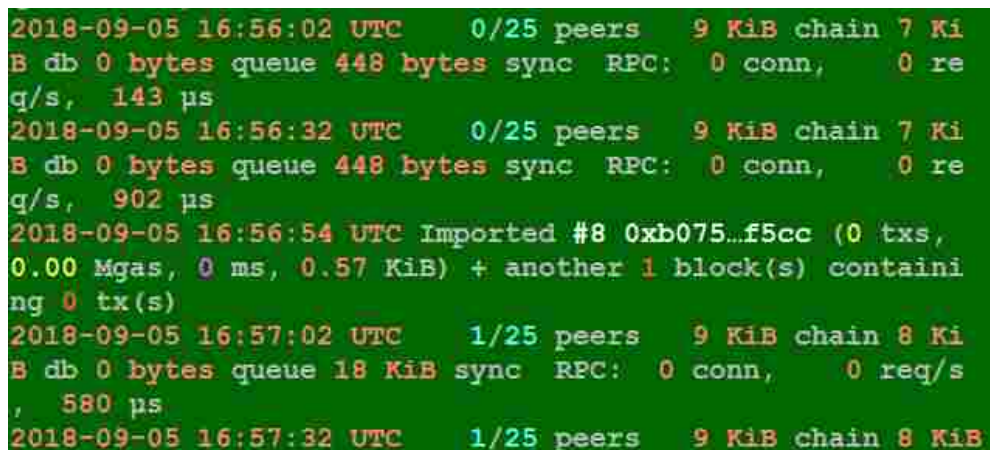
```
curl --data '{
    "jsonrpc": "2.0",
    "method": "parity_addReservedPeer",
    "params": ["Add here the result"],
    "id": 0
}' -H "Content-Type: application/json" -X POST
localhost:8541
```

so it will be :

```
curl --data '{
    "jsonrpc": "2.0",
    "method": "parity_addReservedPeer",
    "params": ["enode://24829e7ac2505b63256dbbfcc
```

```
a5e041d84834e9d30c083869fb20da389
b1e3d36d5096a503b9296b785eba5c2a57
dc6af81b8bbb3d601cb72dbe7624e8ea7347
@192.168.56.103:30300"],
"id":0
}'-H "Content-Type: application/json"
-X POST
localhost:8541
```

As shown in Fig. 4.16, the nodes should indicate from 0/25 to 1/25 peers in the terminal in which both of the nodes are connected.



```
2018-09-05 16:56:02 UTC    0/25 peers    9 KiB chain 7 Ki
B db 0 bytes queue 448 bytes sync RPC: 0 conn, 0 re
q/s, 143 µs
2018-09-05 16:56:32 UTC    0/25 peers    9 KiB chain 7 Ki
B db 0 bytes queue 448 bytes sync RPC: 0 conn, 0 re
q/s, 902 µs
2018-09-05 16:56:54 UTC Imported #8 0xb075...f5cc (0 txs,
0.00 Mgas, 0 ms, 0.57 KiB) + another 1 block(s) containi
ng 0 tx(s)
2018-09-05 16:57:02 UTC    1/25 peers    9 KiB chain 8 Ki
B db 0 bytes queue 18 KiB sync RPC: 0 conn, 0 req/s
, 580 µs
2018-09-05 16:57:32 UTC    1/25 peers    9 KiB chain 8 KiB
```

Figure 4.16: Nodes are connected

4.1.3 Smart contract

After building the environment, the smart contract is written with 167 lines of Solidity code, which is available in Appendix A in Remix Solidity IDE. Solidity language is considered as an object-oriented programming language, which it can be used on any blockchain platform. Furthermore, Remix Solidity IDE is a web browser based IDE that is known as Browser Solidity which allows developers to write and check the functionality of their

smart contracts before deploying and running the code.

4.2 Performance evaluation

This section provides the results of testing the proposed system in two environments, namely the private PoA built for this present work and a public PoA. The proposed smart contract is tested on a public PoA in order to measure the performance of phases on a public network with more than two nodes.

4.2.1 Private network

Section 4.1 described the setup of the testing environment, namely the Ethereum Parity Proof of Authority blockchain network, which consists of the two Parity nodes that are the focus of this test of the exchange between nodes. These are in addition to four accounts, of which two are authority accounts selected as slot leaders to validate transactions and issue blocks and two are user accounts, which serve as Merchant and Supplier, to make fair exchange transactions to the blockchain. Hence, the presented experiments contain some authority nodes and user nodes to deploy the proposed smart contract.

Cost estimates of the block gas limit are simulated or tested in each phase of the proposed smart contract on the testing network, allowing the cost of a transaction that will be validated or processed on the real system to be known. More complex operating of the proposed code will cost more gas. The currency used is Gwei, which is =118.49 Ether. At the time of writing, 1 Ether =118.49 USD.

4.2.2 Public network

The primary objective of the proposed scheme is to devise a fair supply chain system. Implementation of a software prototype was conducted on Ethereum test network. The supply process was depicted by the smart contract. In the experiments, Solidity was

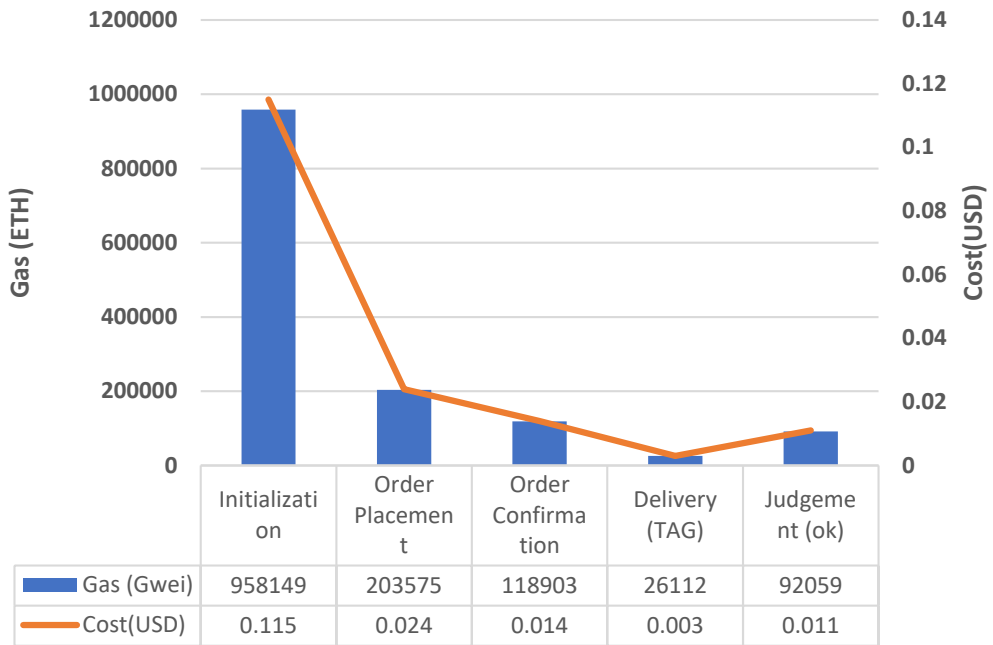


Figure 4.17: Estimates the Gas versus the USD of each phase

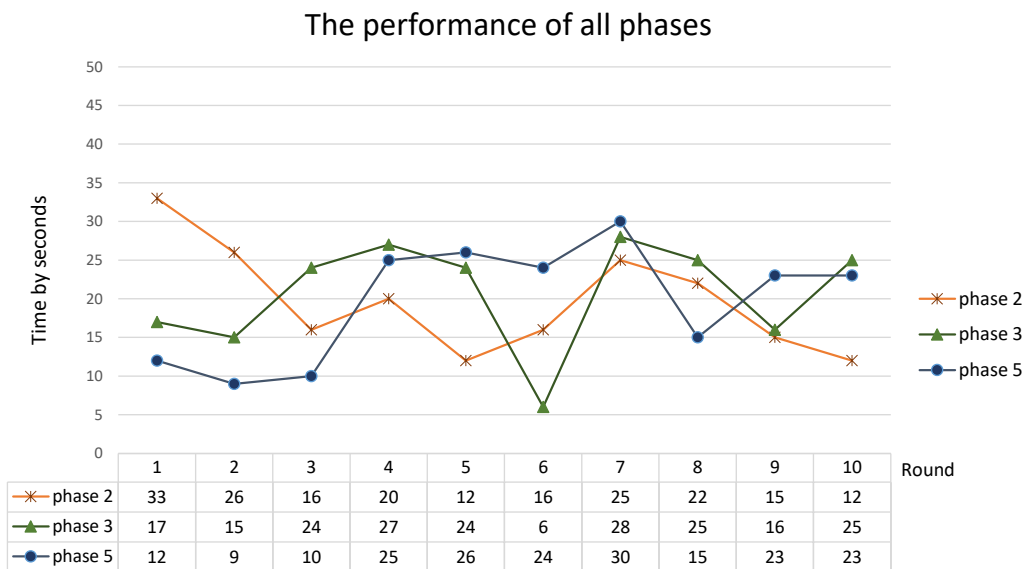


Figure 4.18: Transaction confirmation time in Rinkeby

used to develop smart contracts. The smart contract is saved as Ethereum Virtual Machine (EVM) Bytecode in the blockchain. Once a contract gets stored in the blockchain, it has a unique address so all the participating parties can interact with it using this contract address. JavaScript and JAVA programming language were used in developing the primary logic function in the middle layer. The implementation was performed on the Rinkeby [20], which is part of the official Ethereum public test network. The daily number of transactions is approximately 60,000 more than 10,000 smart contracts are currently deployed to Rinkeby. It is a “Proof-of-Authority” network, meaning that signing of the blocks is done by well-known and trusted members of the community [39]. This process prevents any attacker from hijacking the mining power within the network. In order to compare the performances given different numbers of mining nodes and transactions, this current study repeated experiments in Rinkeby for 10 times, which focuses mainly on two issues: aspects of network latency as well as synchronous messages in the blockchain. The first issue refers to the time of transaction confirmation, including sending and conformation of the order as well as the judge phase. The purpose is to verify whether the transactions on the supply chain are in synchrony within the entire network and also within a time that is acceptable. The mean block time taken during the mining in Rinkeby is around 15 seconds. These experiments comprised 10 transactions that each of them was mined into a single block. The assigned block numbers range from a value of 2,267,171 to 2,271,710. The mean gas usage for each of the transactions is around 300,000. As displayed in Fig, the mean time taken for transaction confirmation on sending and conforming an order is between 6 to 33 seconds. Each of the transactions is confirmed within around 2 block time. With regard to the low or high points in Fig 4.18, location of the special block is determined and positive correlation between the time of transaction confirmation and the number of transactions within a single block is identified. As an example, there is an apparent high point in the first experiments for phase 2 and the presence of 11 transactions is discovered in the same block. This number is far higher compared with the mean transactions in the same block.

Chapter 5

Conclusions and Future Works

5.1 Summary of research

In presenting a fully complete system that utilizes blockchain for the supply chain, the presented work begins with a discussion of issues and methods in general supply chain management that have been in existence over the last decade, while focusing on the lack of fairness that can occur between all involved parties. This overview is followed by an introduction to a blockchain-based supply chain management system for IIoT that considers how the features of this technology can fill the gaps in the existing system. Specific emphasis is placed on the realization of a fairness protocol in the smart contract. As a consequence, fairness property for each side is defined and a process for how to achieve fairness in the proposed scheme is described. Furthermore, security analysis and experiments are conducted as part of this work in order to illustrate the feasibility and efficiency of the proposed scheme.

5.2 Future research directions

Addressing the issues that concern all involved parties in the field of supply chain management offers further potential for future research. In particular, investigating the tracking

of goods by satellite, especially for global trade, would be valuable in cases of dispute, and would help to more widely achieve the concept of transparency.

Appendix A

Codes

Listing A.1: Solidity language

```
pragma solidity ^0.4.0;
// This contract is to realize the fairness goods exchange
// between seller and buyer.
contract IIoTChain {
    address public buyer;
    address public seller;
    // alpha
    uint public radio;
    // this time refers to interval between confirmation and
    // delivery
    uint public T1;
    // this time refers to the entity to upload evidence (proof of
    // misbehavior)
    uint public T2;
    mapping(address => uint) accountBalance;
    mapping(address => uint) accountDeposit;
    mapping(string => Item) itemList;
    Order[] orderList;
    uint256 index = 0;
    // this struct is used to describe the product
    struct Item {
        string name;
        uint price;
    }
}
```

```

        bool isAvailable;
    }
    struct Order {
        Item item;
        uint orderDate;
        uint number;
        uint deliveryDate;
        uint confirmedDate;
        uint deadline;
        Status status;
    }
    enum Status {
        initialized, // 0
        Confirmed, // 1
        Accepted, // 2
        Completed //
    }
    function addItem(string _name, uint _price) public {
        //only seller can add item list
        require(msg.sender == seller);

        itemList[_name].name = _name;
        itemList[_name].price = _price;
        itemList[_name].isAvailable = true;
    }
    function setItemOrderStatus(string _name,bool _isAvailable)
        public {
        //only seller can add item list
        require(msg.sender == seller);

        itemList[_name].isAvailable = _isAvailable;
    }
    // this function is used for initialization
    function IIoTChain(uint _T1, uint _T2, uint _radio, address
        _buyer, address _seller) public {
        T1 = _T1;
        T2 = _T2;
        radio = _radio;
    }

```

```

    buyer = _buyer;
    seller = _seller;
}
//this function is for buyer to make the order
function orderPhase(string _itemName, uint number, bytes32
    orderSignature, uint _deadline, uint8 v, bytes32 r,
    bytes32 s)
public payable {
    //check the identification of buyer
    require(msg.sender == buyer);
    if(itemsList[_itemName].isAvailable == false) return;
    //check the signature of buyer
    require(isSigned(buyer, orderSignature, v, r, s));
    Order memory newOrder = orderList[index++];
    newOrder.item = itemsList[_itemName];
    newOrder.orderDate = now;
    newOrder.number = number;
    newOrder.deadline = _deadline;
    newOrder.status = Status.initialized;
    uint totalPrice = number * itemsList[_itemName].price;
    //check the balance of the account
    require(msg.value >= totalPrice * (1 + radio/100));
    // buyerDeposit
    setDeposit(totalPrice * (1 + radio/100));
}
function orderConfirmPhase(string _prodectName, uint256
    _index, bytes32 confirmSignature, uint8 v, bytes32 r,
    bytes32 s) public
payable {
    //check the identification of seller
    require(msg.sender == seller);
    //check the signature of buyer
    require(isSigned(seller, confirmSignature, v, r, s));
    //Items storage itemsTmp;
    Order memory order = orderList[_index];
    order.confirmedDate = now;
    //check the deposit

```

```

uint totalDeposit = order.number *
    itemList[_prodectName].price * (radio/100);
require(msg.value > totalDeposit);
// sellerDeposit
setDeposit(totalDeposit);
order.status = Status.Confirmed;
}
function generateTags(uint256 _index) public returns (string
tags) {
Order memory order = orderList[_index];
string memory tagsA = strConcat(order.item.name, new
string(order.deliveryDate));
return tagsA;
}
function acceptOrderPhase(string _prodectName, uint256 _index,
bytes32 acceptSignature,uint8 v, bytes32 r, bytes32 s)
public
payable {
//check the idetification of seller
require(msg.sender == buyer);
//check the signature of buyer
require(isSigned(buyer,acceptSignature,v,r,s));
//check the balance of the account
require((now - orderList[_index].confirmedDate) < T1);
orderList[_index].status = Status.Accepted;
}
function evidenceGenerate(uint256 _index, string evidence,
bytes32 evidenceSignature,uint8 v, bytes32 r, bytes32 s)
public {
Order memory order = orderList[_index];
require(now > (order.deliveryDate + T1) && now <
(order.deliveryDate + T1 + T2));
require(isSigned(msg.sender,evidenceSignature,v,r,s));
if(msg.sender == buyer){
withdraw(order.number * order.item.price * (1 +
radio/100));
seller.transfer(order.item.price * (radio/100));
} else if (msg.sender == seller) {

```

```

        withdraw(order.number * order.item.price * (1+
            radio/100));
        buyer.transfer(order.item.price * (radio/100));
    } else {
        return;
    }
}
// this function is to do the payment
function orderPaymentPhase(uint256 _index) public
payable {
    Order memory order = orderList[_index];
    require(now > (order.deliveryDate + T1 + T2));
    //proof of misbehavior function by a seller;
    if(msg.sender == buyer){
        withdraw(order.number * order.item.price * (radio/100));
    } else if (msg.sender == seller) {
        withdraw(order.number * order.item.price * (1+
            radio/100));
    } else {
        return;
    }
    order.status = Status.Completed;
}
//this function is for each entity to make a deposit
function setDeposit(uint amt)
{
    accountDeposit[msg.sender] += amt;
}
function withdraw(uint amt)
{
    uint withdrawAmount = amt;
    uint newBalance = accountDeposit[msg.sender] -
        withdrawAmount;
    accountBalance[msg.sender] = newBalance;
}
function recoverAddr(bytes32 msgHash, uint8 v, bytes32 r,
    bytes32 s) returns (address) {
    return ecrecover(msgHash, v, r, s);
}

```

```
}
function isSigned(address _addr, bytes32 msgHash, uint8 v,
    bytes32 r, bytes32 s) returns (bool) {
    return ecrecover(msgHash, v, r, s) == _addr;
}
function strConcat(string a, string b) internal returns
    (string){
    bytes memory ba = bytes(a);
    bytes memory bb = bytes(b);
    string memory ret = new string(ba.length + bb.length);
    bytes memory bret = bytes(ret);
    uint k = 0;
    for (uint i = 0; i < ba.length; i++)bret[k++] = ba[i];
    for (uint j = 0; j < bb.length; j++) bret[k++] = bb[j];
    return string(bret);
}
}
```

Bibliography

- [1] Supply chain industry: biggest challenges 2018. <https://www.statista.com/statistics/829634/biggest-challenges-supply-chain/>.
- [2] Saveen A Abeyratne and Radmehr P Monfared. Blockchain ready manufacturing supply chain using distributed ledger. 2016.
- [3] Louise Axon. Privacy-awareness in blockchain-based pki. 2015.
- [4] Mohamed Baza, Mahmoud Nabil, Muhammad Ismail, Mohamed Mahmoud, Erchin Serpedin, and Mohammad Rahman. Blockchain-based privacy-preserving charging coordination mechanism for energy storage units. *arXiv preprint arXiv:1811.02001*, 2018.
- [5] Nikola Bozic, Guy Pujolle, and Stefano Secci. A tutorial on blockchain and applications to secure network control-planes. In *2016 3rd Smart Cloud Networks & Systems (SCNS)*, pages 1–8. IEEE, 2016.
- [6] Richard G Brown. Introducing r3 cordatm: A distributed ledger designed for financial services. *R3CEV blog*, 2016.
- [7] Vitalik Buterin. On public and private blockchains. *Ethereum blog*, 7, 2015.
- [8] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.
- [9] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, volume 310, 2016.
- [10] Christian Cachin and Jan Camenisch. optimistic fair secure computation. In *Advances in Cryptology (CRYPTO 2000)*. Springer Science & Business Media, 2000.

- [11] Injazz J Chen and Anthony Paulraj. Understanding supply chain management: critical research and a theoretical framework. *International Journal of production research*, 42(1):131–163, 2004.
- [12] Sungchul Choi and Paul R Messinger. The role of fairness in competitive supply chain relationships: An experimental study. *European Journal of Operational Research*, 251(3):798–813, 2016.
- [13] David Lee Kuo Chuen. *Handbook of digital currency: Bitcoin, innovation, financial instruments, and big data*. Academic Press, 2015.
- [14] Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.
- [15] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 967–984. ACM, 2018.
- [16] Ittay Eyal. Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities. *Computer*, 50(9):38–49, 2017.
- [17] Shafi Goldwasser. How to play any mental game, or a completeness theorem for protocols with an honest majority. *Proc. the Nineteenth Annual ACM STOC'87*, pages 218–229, 1987.
- [18] Dominique Guegan. Public blockchain versus private blockchain. 2017.
- [19] Dirk Hahnel, Wolfram Burgard, Dieter Fox, Ken Fishkin, and Matthai Philipose. Mapping and localization with rfid technology. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 1015–1020. IEEE, 2004.
- [20] Kedar Iyer and Chris Dannen. The ethereum development environment. In *Building Games with Ethereum Smart Contracts*, pages 19–36. Springer, 2018.
- [21] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.
- [22] Henning Kagermann, Johannes Helbig, Ariane Hellinger, and Wolfgang Wahlster. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*. Forschungsunion, 2013.

- [23] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*, 2017.
- [24] Kari Korpela, Karri Mikkonen, Jukka Hallikas, and Mikko Pynnönen. Digital business ecosystem transformation—towards cloud integration. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3959–3968. IEEE, 2016.
- [25] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, 2016.
- [26] Yong Ming Kow and Xianghua Ding. Hey, i know what this is!: Cultural affinities and early stage appropriation of the emerging bitcoin technology. In *Proceedings of the 19th International Conference on Supporting Group Work*, pages 213–221. ACM, 2016.
- [27] Alptekin Küpçü and Anna Lysyanskaya. Usable optimistic fair exchange. In *Cryptographers Track at the RSA Conference*, pages 252–267. Springer, 2010.
- [28] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.
- [29] Meng Li, Liehuang Zhu, and Xiaodong Lin. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet of Things Journal*, 2018.
- [30] Ming Li, Jian Weng, Anjia Yang, Wei Lu, Yue Zhang, Lin Hou, Jia-Nan Liu, Yang Xiang, and Robert Deng. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*, 2018.
- [31] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017.
- [32] Xiaodong Lin, Johnny W Wong, and Weidong Kou. Performance analysis of secure web server based on ssl. In *International Workshop on Information Security*, pages 249–261. Springer, 2000.

- [33] Dongxiao Liu, Amal Alahmadi, Jianbing Ni, Xiaodong Lin, et al. Anonymous reputation system for iiot-enabled retail marketing atop pos blockchain. *IEEE Transactions on Industrial Informatics*, 2019.
- [34] Ronald Maier, Giuseppina Passiante, and Shujun Zhang. Creating value in networks. *International Journal of Innovation and Technology Management*, 8(03):357–371, 2011.
- [35] Robert M Monczka, Robert B Handfield, Larry C Giunipero, and James L Patterson. *Purchasing and supply chain management*. Cengage Learning, 2015.
- [36] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
- [37] Charles Noyes. Bitav: Fast anti-malware by distributed blockchain consensus and feedforward scanning. *arXiv preprint arXiv:1601.01405*, 2016.
- [38] Andrea Ordanini and Gaia Rubera. Strategic capabilities and internet resources in procurement: A resource-based view of b-to-b buying process. *International Journal of Operations & Production Management*, 28(1):27–52, 2008.
- [39] Abdallah Zoubir Ourad, Boutheyna Belgacem, and Khaled Salah. Using blockchain for iot access control and authentication management. In *International Conference on Internet of Things*, pages 150–164. Springer, 2018.
- [40] Fei Qin, Feng Mai, Michael J Fry, and Amitabh S Raturi. Supply-chain performance anomalies: Fairness concerns under private cost information. *European Journal of Operational Research*, 252(1):170–182, 2016.
- [41] Pamela Vagata and Kevin Wilfong. Scaling the facebook data warehouse to 300 pb. *Facebook Code, Facebook*, 10, 2014.
- [42] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.
- [43] Wei Wang, Kah Chan Teh, and Kwok Hung Li. Artificial noise aided physical layer security in multi-antenna small-cell networks. *IEEE Transactions on Information Forensics and Security*, 12(6):1470–1482, 2017.
- [44] Wei Wang, Kah Chan Teh, Kwok Hung Li, and Sheng Luo. On the impact of adaptive eavesdroppers in multi-antenna cellular networks. *IEEE Transactions on Information Forensics and Security*, 13(2):269–279, 2018.

- [45] J Weng, Jian Weng, J Zhang, M Li, Y Zhang, and W Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *Cryptology ePrint Archive, Report 2018/679*, 2018.
- [46] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.
- [47] A. Yang, E. Pagnin, A. Mitrokotsa, G. P. Hancke, and D. S. Wong. Two-hop distance-bounding protocols: Keep your friends close. *IEEE Transactions on Mobile Computing*, 17(7):1723–1736, July 2018.
- [48] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.
- [49] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.