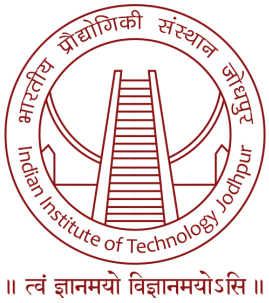
Cutting-Edge Consensus Algorithms in Scalable and Interoperable Blockchain

A Thesis Submitted By
Lokendra Vishwakarma

in partial fulfillment of the requirements for the award of the degree of
Doctor of Philosophy



Indian Institute of Technology Jodhpur
Computer Science And Engineering
April 2024

Declaration

I hereby declare that the work presented in this thesis titled *Cutting-Edge Consensus Algorithms in Scalable and Interoperable Blockchain* submitted to the Indian Institute of Technology Jodhpur in partial fulfillment of the requirements for the award of the degree of *Doctor of Philosophy*, is a bonafide record of the research work carried out under the supervision of Dr. Debasis Das. The contents of this thesis, in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Lokendra Vishwakarma
P19CSE012
Computer Science and Engineering
Indian Institute of Technology Jodhpur

Certificate

This is to certify that the thesis titled *Cutting-Edge Consensus Algorithms in Scalable and Interoperable Blockchain*, submitted by Lokendra Vishwakarma (P19CSE012) to the Indian Institute of Technology Jodhpur for the award of the degree of *Doctor of Philosophy*, is a bonafide record of the research work done by him under my supervision. To the best of our knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Debasis Das
Ph.D. Thesis Supervisor
Computer Science and Engineering
Indian Institute of Technology Jodhpur

Abstract

The development of consensus algorithms is crucial for enhancing the scalability and interoperability of blockchain technology, which is key to its wider adoption. However, existing consensus algorithms, including Proof-of-Work (PoW), Proof-of-Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT), face challenges such as high consensus delay, low throughput, and limited transaction scalability, primarily due to their high computational and communication complexity. Moreover, these algorithms rely on third parties to communicate across diverse blockchain platforms, making blockchain vulnerable, which emphasizes the necessity for interoperable solutions. In addition, ensuring information privacy within blockchain systems remains a significant challenge.

Therefore, in this thesis, we develop novel consensus algorithms while focusing on improving consensus delay and throughput. The proposed modified Practical Byzantine Fault Tolerance (mPBFT) and Lightweight Consensus Algorithm (LCA) consensus algorithms are the improvement over the PBFT consensus protocol, which reduces the computation and communication complexity. The GridChain (a reputation based consensus algorithm) and Weighted Node Selection Byzantine Fault Tolerance (WNS-BFT) are the reputation and trust-based consensus algorithms that establish trust among the nodes while keeping the performance high. Additionally, to improve the performance further, we have developed a technique called BlockTree (a nonlinear structured, scalable, and distributed ledger) and UnoShard (Harmonizing Scalability and Security) schemes that address the issue of limited scalability by enabling the parallel processing of transactions. Furthermore, to address the interoperability issues between the diverse blockchain platforms, we developed a technique called CrossLedger (a pioneer cross-chain asset transfer protocol) that eliminates the presence of third parties for communication. This thesis has also addressed the issue of information privacy in the proposed technique called zk-SGB, a privacy-preserved blockchain framework for energy trading in a smart grid using zero-knowledge-proof (ZKP) without compromising system integrity.

Next, this thesis discusses the various applications based on blockchain technology, such as the Internet of Things (IoT), Intelligent Transportation Systems (ITS), and Energy Trading in the Smart Grid. We have developed three schemes for IoT using smart contracts. The first scheme is a BLISS (Blockchain-based Integrated Security System for Internet of Things (IoT) Applications), and the second scheme is SCAB-IoTA (Secure Communication and Authentication for IoT Applications using Blockchain. We have implemented these schemes on the Ethereum blockchain testnet. Additionally, we have developed a testbed using Raspberry Pi devices to show the efficiency of the proposed scheme. Furthermore, we have developed two schemes for ITS. The first scheme is a LBSV (Lightweight Blockchain Security Protocol for Secure Storage and Communication in SDN-Enabled IoV), and the second scheme is SmartCoin (A Novel Incentive Mechanism for Vehicles in Intelligent Transportation System Based on Consortium Blockchain). The LBSV and SmartCoin work on mPBFT and LCA consensus algorithms, respectively. Furthermore, we have also developed three schemes for the smart grid application. The first scheme is a SmartGrid-NG (Blockchain Protocol for Secure Transaction Processing in the Next Generation Smart Grid), the second scheme is an ETradeChain (Blockchain-Based Energy Trading in the Local Energy Market (LEM) Using a Modified Double Auction Protocol), and the third scheme is zk-SGB (Privacy-Preserved Blockchain Framework for Energy Trading in Smart Grid Using Zero Knowledge Proof). The first two schemes are focused on trading and transaction processing and work on GridChain and two-level consensus algorithms, respectively. The zk-SGB is designed to achieve privacy in smart grid transaction processing and works on the WNS-BFT consensus algorithm. A Raspberry Pi 4 Model B devices based testbed has been developed for the implementation of ITS and smart grid schemes.

To My Brave Son Hrishiv,
My Companion and Better Half Nootan, and My Loving Parents.

Acknowledgements

This journey has been a roller-coaster ride for me, filled with many ups and downs, both in my personal life and in my PhD journey. This ride would not have been completed without the many wonderful and amazing people who have supported me and believed in me along the way. With this acknowledgment, I try to express my deepest appreciation and gratitude.

First and foremost, I owe my deepest gratitude to a truly brilliant mind and a very kind soul, my supervisor, **Dr. Debasis Das**. Thank you, Dr. Das, for your untiring support, help, patience, and encouragement throughout my studies, which made this work possible. Your advice and guidance, even when you were away, have been priceless. I have been extremely lucky to have you as my supervisor and friend. Thank you from the bottom of my heart for being there for me whenever needed in my research and beyond. I would like to give special thanks to the Head of the Department, **Prof. Gaurav Harit**, for his valuable support in every aspect of my PhD journey. I am immensely grateful to my student research committee (SRC) professors, **Dr. Suman Kundu**, **Dr. Palash Das**, **Dr. Nilkamal Hazra**, and my former SRC member **Dr. Chiranjoy Chattopadhyay**, for their invaluable guidance, advice, support, and enthusiasm. They were always there to support me and consistently pushed me out of my comfort zone to empower me as a person and researcher.

I would like to express my deepest gratitude to **Prof. Sajal K. Das** (Missouri S&T, USA). His extensive knowledge and unwavering support played a crucial role in shaping my research and refining my thinking. Our conversations were consistently insightful and enjoyable, motivating me to pursue excellence. Prof. Das's wisdom and generosity had a profound influence on both my work and personal development. I would also like to thank to **Prof. Christian Becker** (University of Stuttgart) **Prof. Sukumar Nandi** (IIT Guwahati, India) and **Prof. Athanasios V. Vasilakos** (Lulea University of Technology, Sweden) for the collaborative work. Your feedback and guidance provide me with a great sense of knowledge and clarity. In the last leg of my PhD journey, I met with **Dr. Nitin Awathare**, who taught me to think about the research problem from different directions and different perspectives. I am also very thankful to him for his support and guidance, which broadened my thinking in blockchain research.

I believe that no journey is complete without friends. In this journey, my friends have been my pillars of strength, both emotionally and spiritually. To my dear friends **Himani, Ankur Nahar**, and **Aashish Bohra**, your unwavering support has been a beacon of light throughout this journey. Your friendship has been a true blessing, and I am deeply grateful for your presence in my life. This journey became even more special with the spiritual engagement due to my friend **Jayant Vyas**. From being a colleague in Indore to becoming a special friend at IIT Jodhpur, your friendship has been life-changing for me. **Amritesh Kumar**, you are truly a pure soul, and your support has been invaluable. You have always been there for me, offering support beyond just the PhD work. **Monu Nagar, Arju**, our camaraderie has been a source of resilience. Together, we have faced challenges, celebrated victories, and created memories that will last a lifetime. I also extend my heartfelt thanks to every member of the VANET lab; **Bhumika, Umang, Shristy, Samanyu**, and **Koustav** for their support and engaging discussions throughout this journey.

Without the support of family, none of this would have been possible. I am deeply thankful to my father **Mr. Ratan Lal Vishwakarma** and mother **Ms. Sunita Vishwakarma** for believing in me and my dreams. You encouraged me to pursue my goals and passions. There are not enough words to express my gratitude toward you. To my wife **Ms. Nootan**, you are truly my life companion. You were more worried during all

my failures, and you cheered for me and celebrated my tiniest achievements. We have shared many difficult times, and you stayed strong and motivated me to go beyond my limits.

Last but not least, to my brave son **Harshiv**, your arrival in my life brought a ray of hope to do better. Your smile and laughter suppressed all the stress of this journey. Harshiv, thank you for being my brightest star and ray of hope; this achievement is dedicated to you, my brave boy.

Lokendra Vishwakarma
P19CSE012
Computer Science and Engineering
Indian Institute of Technology Jodhpur

Table of Contents

Declaration I

Certificate III

Abstract V

Acknowledgments IX

Table of Contents XI

List of Figures XV

List of Table XVII

1 Introduction 1
- 1.1 Underline Blockchain Terminology . . . 1
- 1.2 Consensus Algorithms . . . 3
  - 1.2.1 Proof Based Consensus Algorithms . . . 3
  - 1.2.2 Vote Based Consensus Algorithm . . . 5
- 1.3 Smart Contracts . . . 7
- 1.4 Type of Blockchain . . . 8
  - 1.4.1 Public Blockchain . . . 8
  - 1.4.2 Private Blockchain . . . 8
  - 1.4.3 Consortium Blockchain . . . 9
- 1.5 Challenges in Blockchain . . . 9
  - 1.5.1 Scalability . . . 10
  - 1.5.2 Interoperability . . . 10
  - 1.5.3 Privacy . . . 11
  - 1.5.4 Vulnerabilities in Blockchain . . . 11
- 1.6 Blockchain Applications . . . 13
  - 1.6.1 Internet of Things (IoT) . . . 13
  - 1.6.2 Intelligent Transportation Systems (ITS) . . . 14
  - 1.6.3 Smart Grid . . . 14
- 1.7 Motivation and Contributions . . . 14
- 1.8 Thesis Structure . . . 16

2 Related Work 17
- 2.1 Consensus Algorithm Enhancements . . . 17
- 2.2 State-of-the-art Schemes for Scalable Blockchain . . . 18

2.3 State-of-the-art Schemes for Interoperable Blockchain . . . 21
2.4 State-of-the-art Schemes for Blockchain Application . . . 23

3 Scalable Blockchain **27**
3.1 Challenges in Achieving Scalability . . . 27
3.2 Blockchain Sharding . . . 28
3.3 Contributions . . . 29
3.4 BlockTree: A Nonlinear Structured, Scalable Distributed Ledger . . . 30
3.4.1 Sharding in BlockTree . . . 31
3.4.2 Transaction Sharding . . . 31
3.4.3 Leader Selection, Leader Block, and Data Block . . . 32
3.4.4 Remuneration Model . . . 36
3.5 UnoShard: Harmonizing Scalability and Security . . . 36
3.5.1 Shard Allocation . . . 37
3.5.2 Intra-Shard Consensus . . . 38
3.5.3 Cross-Shard Consensus . . . 40
3.6 Performance Analysis . . . 45
3.6.1 Simulation Environment . . . 46
3.6.2 Consensus Delay . . . 46
3.6.3 Throughput . . . 47
3.6.4 Fairness . . . 47
3.6.5 Block Congestion Control . . . 48
3.7 Security Analysis . . . 49
3.7.1 Prevention of Double Spending . . . 51
3.7.2 Illegitimate Fork Prevention . . . 51
3.7.3 Scalability Enhancement . . . 52

4 Interoperability **53**
4.1 Challenges in Interoperable Blockchains . . . 53
4.2 *CrossLedger*: A Pioneer Cross-Chain Asset Transfer Protocol . . . 54
4.2.1 System Model . . . 55
4.2.2 Background . . . 56
4.2.3 Methodology . . . 61
4.2.4 Essential Characteristics of Interoperable Blockchahin . . . 65
4.3 Security Analysis . . . 67

5 Blockchain Applications **71**
5.1 Solution for IoT Application . . . 72
5.1.1 Motivation and Contribution . . . 72
5.1.2 System Model . . . 73
5.1.3 Adversary Model . . . 74
5.1.4 BLISS: Blockchain-based Integrated Security System for Internet of Things (IoT) Applications . . . 75
5.1.5 SCAB-IoTA: Secure Communication and Authentication for IoT Applications Using Blockchain . . . 81
5.1.6 Implementation Detail and Performance Analysis . . . 85
5.1.7 Security Analysis . . . 89

5.2 Solution for ITS Applications . . . 94
5.2.1 Motivation and Contribution . . . 95
5.2.2 Network Model . . . 96
5.2.3 Assumptions . . . 98
5.2.4 Adversary Model . . . 99
5.2.5 LBSV: Lightweight Blockchain Security Protocol for Secure Storage and Communication in SDN-enabled IoV . . . 100
5.2.6 SmartCoin: A Novel Incentive Mechanism for Vehicles in Intelligent Transportation System Based on Consortium Blockchain . . . 105
5.2.7 Implementation Environment and Performance Analysis . . . 108
5.2.8 Security Analysis . . . 112
5.3 Blockchain-based Secured Smart Grid System . . . 117
5.3.1 Motivation and Contribution . . . 118
5.3.2 Basic Terminologies in Smart Grid . . . 119
5.3.3 SmartGrid-NG: Blockchain Protocol for Secure Transaction Processing in Next Generation Smart Grid . . . 120
5.3.4 ETradeChain: Blockchain-Based Energy Trading in Local Energy Market (LEM) using Modified Double Auction Protocol . . . 128
5.3.5 Implementation Detail and Performance Analysis . . . 137
5.3.6 Security Analysis for SmartGrid-NG and ETradeChain . . . 144
5.4 Privacy-Preserving Energy Trading in Smart Grid . . . 148
5.4.1 Techniques Used to Provide Privacy in Blockchain . . . 149
5.4.2 *zk-SNARK* Background . . . 149
5.4.3 Public Traceability . . . 150
5.4.4 *zk-SGB*: Privacy-Preserved Blockchain Framework for Energy Trading in Smart Grid Using Zero Knowledge Proof . . . 150
5.4.5 Consumer and Prosumer Communication Based on zk-SNARK . . . 151
5.4.6 WNS-BFT: Consensus Algorithm . . . 152
5.4.7 Performance Analysis . . . 156
5.4.8 Security Analysis of zk-SGB . . . 158

6 Conclusion and Future Work **163**

Publications **164**

References **169**

List of Figures

1.1 Typical graphical representation of blockchain . . . 2
1.2 Working mechanism of PoW . . . 4
1.3 Working mechanism of PoS . . . 5
1.4 Working mechanism of PBFT . . . 6
1.5 Fork in blockchain . . . 12

3.1 An example of a sharded blockchain with three shards. Each shard has its own local ledger L_1, L_2, L_3, and one global ledger. . . . 28
3.2 Struture of BlockTree . . . 31
3.3 Types of blocks and their arrangement . . . 34
3.4 Intra-shard consensus process of UnoShard based on PBFT . . . 40
3.5 This figure illustrates the generation of local blocks (LB_i) and VoteChain of VoteBlocks (VB_j). A Shard Coordinator (SC) is elected in each shard, as depicted here. . . . 44
3.6 Comparative analysis of BlockTree with state-of-the-art schemes based on the (a) consensus delay and (b) throughput . . . 47
3.7 Comparative analysis of BlockTree with state-of-the-art schemes based on (a) fairness and (b) block congestion control . . . 49

4.1 *CrossLedger*'s asset transfer mechanism between two blockchains, source blockchain A and destination blockchain B . . . 56
4.2 Essential parameter exchange between AF_A and AF_B . . . 62
4.3 Sequence for asset transfer communication from blockchain A to blockchain B . . . 65

5.1 System model for IoT application . . . 74
5.2 Device authentication process in BLISS . . . 76
5.3 Candidate-to-candidate secure communication in BLISS . . . 79
5.4 Cluster creation and device authentication . . . 83
5.5 Secret-key share and M2M communication . . . 84
5.6 Testbed for one cluster of 5 Raspberry Pi model B devices . . . 86
5.7 Metamask interface to connect with Ethereum testnet . . . 87
5.8 Output of the JoinCluster contract. The function call isAvailableDeviceID shows the device is found, mentioned under the decoded output . . . 88
5.9 A basic network model of proposed ITS solution . . . 97
5.10 Representation of TPD with computation of different parameters. . . . 98
5.11 (a) Initialization phase of LBSV, (b) Registration phase of LBSV . . . 101
5.12 Communication phase of LBSV . . . 104
5.13 Communication between Vehicle, RSU, and Controller Node . . . 108
5.14 (a) Storage and communication cost of SmartCoin (SC) and LBSV with state-of-the-art scheme (b) Consensus delay of SmartCoin (SC) and LBSV with state-of-the-art scheme . . . 111

5.15 Computational efficiency of SDN . . . 112
5.16 Fault tolerance in SmartCoin . . . 115
5.17 SmartGrid-NG system architecture . . . 121
5.18 ETradeChain system model . . . 129
5.19 Testbed setup for SmartGrid-NG using one computer system and eight Raspberry Pi boards . 138
5.20 Implementation environment for ETradeChain testbed . . . 139
5.21 ETradeChain working process on the RP4 devices. These are the command line interface of RP4 doing all the required processing . . . 140
5.22 Comparative analysis of GridChain based on (a) consensus delay and (b) throughput (number of transactions executed per second) with state-of-the-art schemes . . . 141
5.23 Comparative analysis of ETradeChain based on (a) consensus delay and (b) throughput with state-of-the-art schemes . . . 143
5.24 Sequence diagram for communication between the consumer (smart meter) and prosumer (consensus node) in *zk-SGB* . . . 152
5.25 Comparative analysis of *zk-SGB* based on (a) consensus delay and (b) throughput with state-of-the-art schemes . . . 158

List of Tables

1.1 Comparative analysis between Public, Private, and Consortium blockchains 9

2.1 Comparison of consensus algorithm enhancements 19
2.2 Comparison of state-of-the-art schemes for scalable blockchain 22
2.3 Comparison of state-of-the-art schemes for interoperable blockchain 24

5.1 Symbol table for BLISS and SCAB-IoTA 74
5.2 Individual computation time of different cryptographic functions for IoT testbed 88
5.3 Computation cost (CC) and energy consumption (EC) of BLISS, SCAB-IoTA, compared with state-of-the-art schemes on Desktop and Raspberry Pi (RP) 89
5.4 Comparative storage and communication cost of BLISS, SCAB-IoTA with state-of-the-art schemes 89
5.5 Comparative security analysis of BLISS and SCAB-IoTA with state-of-the-art schemes . . . 90
5.6 Symbol table for LBSV and SmartCoin 98
5.7 Types of messages and the corresponding SmartCoins 106
5.8 Individual computation time of different cryptographic functions in the testbed environment 110
5.9 Computation cost (CC) and energy consumption (EC) of LBSV and SmartCoin compared with state-of-the-art schemes on Desktop and Raspberry Pi (RP) 110
5.10 Comparative storage and communication cost of LBSV and SmartCoin with state-of-the-art schemes 111
5.11 Comparative security analysis of LBSV and SmartCoin with state-of-the-art schemes based on different attacks 117
5.12 Mathematical symbol table for SmartGrid-NG and ETradeChain 121
5.13 Individual computation time of different cryptographic functions for smart grid testbed . . . 141
5.14 Performance analysis of SmartGrid-NG based on the comparison of computation cost (CC) in milliseconds and energy consumption (EC) in Joule on Desktop and Raspberry Pi 4 2GB (RP) with state-of-the-art schemes 142
5.15 Performance analysis of ETradeChain based on the comparison of computation cost (CC) in milliseconds and energy consumption (EC) in Joule of ETradeChain on Raspberry Pi 4 (RP) 2GB and 4GB with state-of-the-art schemes 143
5.16 Comparison of storage and communication cost of ETradeChain with state-of-the-art schemes 144
5.17 Security comparisons of SmartGrid-NG and ETradeChain with state-of-the-art schemes . . 144
5.18 Comparison of computation cost (CC) in milliseconds and energy consumption (EC) in Joule of *zk-SGB* on Raspberry Pi 4 (RP) 2GB and 4GB with state-of-the-art schemes 159

1

Introduction

Blockchain has emerged as a prominent area of research amongst researchers and academicians after the huge success of Bitcoin cryptocurrency [1]. The distinguished feature of blockchain is that it enables a fully decentralized system, meaning no single entity (node) governs or controls the blockchain operations [2]. Apart from decentralization, blockchain also offers an immutable, transparent, and reliable distributed ledger on the ordered set of transactions by leveraging cryptography and carefully designed incentive schemes [3]. Blockchain achieved all these features, i.e., decentralization, immutability, transparency, and distributed ledger using the concept of consensus algorithms [4]. As blockchain technology matured, researchers sensed its applicability far beyond digital currencies, leading to programmable blockchains like Ethereum [5]. In the programmable blockchain, a novel concept is introduced called a smart contract [6], a self-triggered program code with the terms of the agreement directly written into the code. This capability opened up possibilities for automating complex processes and creating decentralized applications (Dapps) that operate on the blockchain [6]. These developments have significantly broadened blockchain's applicability, positioning it as a foundational technology for various decentralized systems beyond cryptocurrencies [7]. With the capacity to stimulate trustless transactions and its inherent resistance to tampering, blockchain is poised to redefine a host of industries, such as the Internet of Things (IoT), Intelligent Transportation Systems (ITS), and energy distribution in Smart Grid [8]. Its promise lies in creating new, decentralized economic and social infrastructures where direct peer-to-peer interactions are not only possible but also reliable, secure, and efficient. Despite the hype, promising features, and vast applications, blockchain technology still faces many significant challenges, such as inefficient consensus algorithms, limited performance and scalability, absence of interoperability, and concerns about privacy [9]. In this thesis, we aim to address all these challenges. Before a deep dive into the thesis, the next section defines four basic building blocks of blockchain: block, blockchain, node, and Merkle root.

1.1 UNDERLINE BLOCKCHAIN TERMINOLOGY

Blockchain's data structure is ingeniously designed to be both transparent and secure. Each block within the chain contains a header and a body. The header includes the Block Number (B_{ID}), Merkle Root Hash (Mrt), Timestamp (T), Block Hash (BH), and Previous Block Hash (PBH). The body contains lists of individual transactions. Each block is connected to its previous block using the hash cryptographic technique called Secure Hash Algorithm (SHA). The security of the blockchain comes from this cryptographic hash function, which ensures any alteration of transaction data within a block is nearly impossible because it

requires altering all subsequent blocks after that, which is computationally infeasible on a distributed network. Further, public-key cryptography allows users to transact on the blockchain with a pair of keys. A public key serves as an address on the blockchain, and a private key is used to sign transactions, proving ownership of blockchain assets. Typically, blockchain is a linear structure very similar to the linked list, i.e., it stores all the transaction history bunched in the blocks, which are linearly connected. Figure 1.1 shows the blockchain structure.

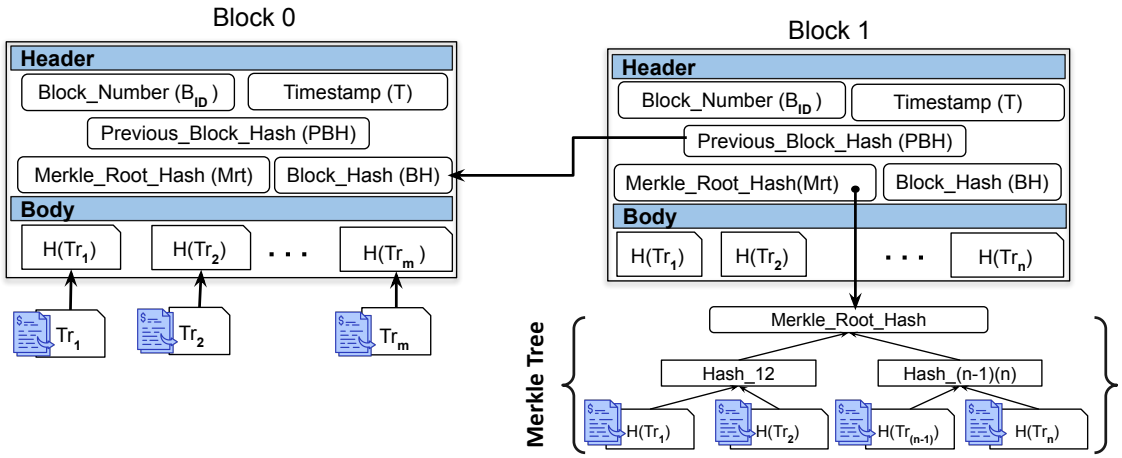


Figure 1.1: Typical graphical representation of blockchain

Definition 1 (Block). *A block is the first and basic component of any blockchain. Here, we denote the block as B. As shown in Figure 1.1, a block is generally a tuple of six elements as* $\sum_B = (B_{ID}, T, BH, PBH, Mrt, \mathbb{S}|_{Tr})$. *The* B_{ID} *is a unique block identifier or number, and* T *is the timestamp. BH and PBH are the block hash and previous block hash, respectively. The next tuple,* Mrt*, is the Merkle root of the set of transactions in the block. The* $\mathbb{S}|_{Tr}$ *is the set of transactions. In cross-chain communication, the transactions include the asset (*$Ast \in \mathbb{S}|_{T}r$*) of users transferred from the source blockchain to the destination blockchain.*

Definition 2 (Blockchain). *A blockchain is the collection of orderly connected blocks in a fixed topological order as* $B_1 \leftarrow B_2 \leftarrow B_3 \leftarrow ... \leftarrow B_N$. *It is a tuple of two elements, i.e.,*$\sum_{Blochain} = (Gns, B_L)$, *where Gns is the genesis block and* $B_L = [B_1, B_2, B_3, ..., B_N]$ *is the list of blocks. The blocks are cryptographically linked in such a way that the hash of* B_1 $(H(B_1))$ *is included in the* B_2*, the hash of* B_2 $(H(B_2))$ *is included in the* B_3*, the hash of* B_{N-1} $(H(B_{N-1}))$ *is included in the* B_N*, as shown in Figure 1.1.*

Definition 3 (Node). *A node is a high-performance processing unit within a distributed network responsible for creating blocks and adding them to the blockchain ledger. These nodes are referred to as the "Miner" nodes. There are two types of nodes in various blockchain technologies, such as Bitcoin. One is a full node, which maintains a complete copy of the blockchain, while the other is a partial node, which relies on the full node to commit transactions or assets.*

Definition 4 (Merkle Root). *The Merkle tree, named after its creator, Ralph Merkle, is a fundamental component of blockchain technology [10]. It encapsulates the summary of all transactions in the block as shown in Figure 1.1. It is a type of binary tree composed of hash-based nodes, i.e., leaf nodes, intermediate nodes, and single root nodes. The root node also referred to as Merkle Root, is denoted as* Mrt*. For a block B containing a set of transactions* $\mathbb{S}|_{Tr} = Tr_1, Tr_2, ..., Tr_n$*, the* Mrt *is computed, where each leaf node represents the hash of a transaction* $H(Tr)$*, each non-leaf node represents the hash of the concatenation of*

its two child nodes, denoted as $H(C_{left}||C_{right})$. This computation proceeds recursively from the leaf nodes to the root of the tree, formally defined as $Mrt = H(...H(H(Tr_1)||H(Tr_2))...||H(...H(Tr_{n-1})||H(Tr_n)))$ for an even number of transactions. The last transaction is duplicated for an odd number of transactions to ensure a complete binary tree. The MerkleRoot serves as a cryptographic fingerprint for all transactions in B, enabling efficient and secure verification of transaction data.

1.2 CONSENSUS ALGORITHMS

The main objective of the consensus algorithm is to solve the byzantine general problem. That is, a system must be able to continue functioning correctly even if some of its nodes fail in arbitrary ways. The Byzantine Generals Problem happens when the members of a network do not agree on the same data values. This could be because some members might be sending out wrong messages on purpose or making mistakes because their systems have crashed or are not working properly. Because of these issues, it's really hard for everyone to come to an agreement.

A consensus algorithm is a critical component of blockchain technology, providing true decentralization. In its simplest form, a consensus algorithm is a general agreement on a single data value (e.g., transactions) among distributed processes or blockchain nodes. These algorithms are necessary for ensuring the reliability and integrity of the transactions despite the fact that some nodes behave maliciously. Thus, before appending a new block in the ledger, the majority of the nodes must agree on the correctness of the block. Various consensus algorithms have been developed for different blockchain platforms. The widely known consensus algorithms are Proof-of-Work (PoW) [1], Proof-of-Stake (PoS) [5], Delegated Proof-of-Stake (DPoS) [11], and Practical Byzantine Fault Tolerance (PBFT) [12]. The PoW, PoS, and DPoS are suitable for permissionless networks, while permissioned networks use different categories of consensus algorithms, such as PBFT. The consensus algorithms are broadly divided into two categories, i.e., proof based consensus algorithm and vote based consensus algorithm [13].

1.2.1 Proof Based Consensus Algorithms

Proof based consensus algorithms are the ones where the participating nodes need to generate some proof or prove the commitment of a certain kind of resource to the network to show that it deserves to create the block and append it to the ledger. Often, the permissionless network adopts this category of consensus algorithms. PoW and PoS are two famous consensus algorithms in this category. In PoW, participating nodes have to solve a highly computational mathematical puzzle to show proof of the correctness of the newly created block. When other nodes in the network agree to the generated proof, the block is added to the ledger [1]. In contrast to the PoW, PoS used the concept of putting a stake as high as possible to become a validator. The node with the highest stake is most likely to be selected as the validator to create the block [14]. Below we have defined these two consensus algorithms in detail.

Proof-of-Work

Bitcoin is the first blockchain platform that used the PoW consensus algorithm to verify and process transactions. The consensus algorithm determines which consensus node (also called the miner) appends the newly created block into the ledger to get the rewards for processing the transactions. Figure 1.2 depicts

the working process of PoW. As shown in the figure, there are two users, A and B, who make transactions that are transferred to the corresponding miners. Following the PoW procedure, one of the miner nodes processes these transactions along with another set of transactions, creates the block, and appends them to the ledger. The PoW prevents miners from immediately appending the new bunch of transactions as a new block on the chain. If this immediate appending block is permitted, many miners could simultaneously grow the chain, making it difficult to determine the ordering of blocks. Instead of the immediate appending block, all the miners must solve the cryptographic puzzle to satisfy the threshold value [1]. Each newly appended block must contain a nonce (a random quantity) to find the block's cryptographic hash that is less than the fixed threshold value. The one-way hash function can not be inverted; due to this property, the only way to find a nonce to satisfy the threshold value is through brute-force search. The miners who first solve the puzzle and win the race append the new block to the chain and are awarded some Bitcoins or other associated cryptocurrency. In the PoW, all the blockchain network peers append the newly generated transactions in the transaction pool and set their status as pending (transactions not included in any of the previous blocks). The miners collect some sets of transactions from the transaction pool that have pending status and create a hash tree to calculate the Merkle root. Miners get the previous block's hash, set the timestamp T, and the nonce equals 0. After setting these parameters, it searches for the hash value with different values of the nonce, which satisfy the threshold hash value. If the hash is found before any other miner, it creates the block using all the parameters, including block hash BH, and broadcasts it to the network for verification. All the other miners receive and verify the block; if it is validated, append it to the ledger.

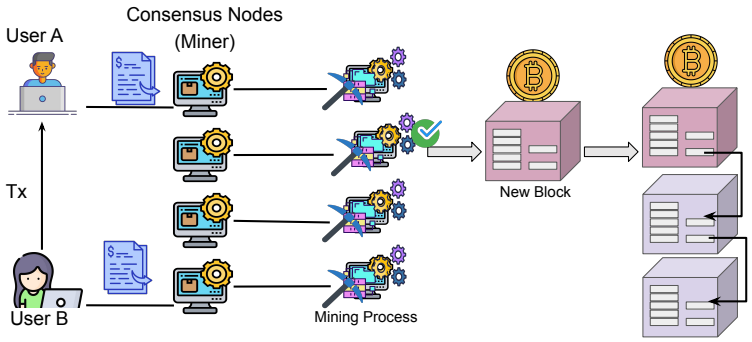


Figure 1.2: Working mechanism of PoW

Proof-of-Stake

A PoS consensus algorithm was proposed to overcome PoW's drawbacks. The PoS also works in the public blockchain, offering protection against attacks. Figure 1.3 shows the typical depiction of the PoS. Similar to PoW, in PoS, there are two users, A and B, who make transactions and transfer to the corresponding consensus nodes. These consensus nodes put some stake in becoming the potential validator for processing transactions and creating new blocks. Based on the highest stake and randomization, one of the nodes is elected as the miner who creates the next block. In PoS, a miner who adds a new block to the blockchain is rewarded as a transaction fee. As opposed to PoW, in PoS, a node becomes a validator by staking cryptocurrency as a security deposit. A new miner is periodically selected as a validator to create and broadcast a new block into the network. All the nodes must submit a security deposit in advance to become a potential validator. The PoS combines the quantity of stakes with one of the two factors to make the selection unbiased and trustworthy for each node on the network. First, coin-age (oldest nodes have the

highest chances to become a new validator), and second, randomization (node which has the lowest hash value and highest stake). In PoS, all the peer nodes put new transactions into the transaction pool. Then, all the nodes who contest to become a validator for the next block raise a stake. This stake is combined with either a coin-age scheme or randomization scheme and node status to select the validator. After getting selected as a validator, the node collects all the pending transactions from the transaction pool, verifies them, creates a block, and broadcasts it to the network for verification. Meanwhile, its stake is locked, and the forging reward (i.e., transaction fees) has not yet been awarded. This step is just a precautionary measure from malicious nodes. If other nodes verify the block, the validator's stake is unlocked, and the reward is awarded. If the coin-age scheme is used for validator selection, then the validator's coin-age is set to 0 for the next round of validator selection so that other nodes also get a chance to become validators, and this makes the scheme fairer to all the nodes. However, if the invalid block is reported as valid, the stake is deducted from the validator and marked as a bad node.

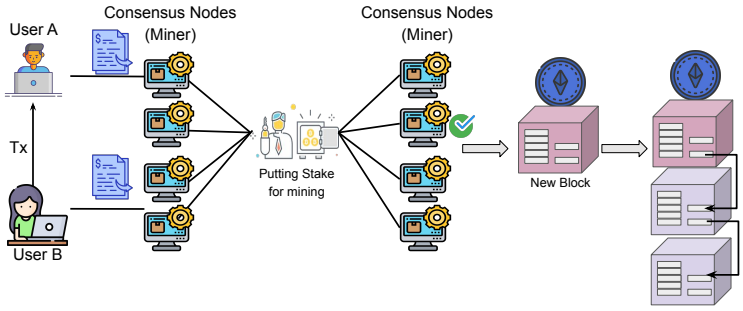


Figure 1.3: Working mechanism of PoS

1.2.2 Vote Based Consensus Algorithm

Unlike proof-based consensus algorithms, Vote based consensus algorithms are the ones that achieve consensus through selection mechanisms, votes, or authority delegation. The selected authority or node creates the block and appends it to the ledger. PBFT and DPoS are the two widely adopted consensus algorithms in this category. PBFT is designed to solve the byzantine general problem. To reach a consensus, PBFT requires a majority (typically $2/3^{rd}$) of the nodes to agree on the state of the currently processing transactions. There are three major stages (i.e., pre-prepare, prepare, and commit) of communication between nodes for the finality of the transactions [12]. In contrast to PBFT, DPoS [11], the stakeholders of the blockchain network elect a number of delegates who are responsible for validating transactions and creating new blocks in the blockchain. DPoS aims to improve the performance by limiting the number of nodes involved in the consensus process. As a result, DPoS has faster transaction processing than PoW and PoS. Below we have defined both these consensus algorithms in detail.

Practical Byzantine Fault Tolerance

Traditional Byzantine Fault Tolerance algorithms were considered to be impractical for real-world applications due to their computational and communication overheads. Castro M. and Liskov B. [12], [15] introduced PBFT as a solution to this problem. Their algorithm was designed to be efficient and practical for real-world systems, ensuring that a distributed service remains available and correct as long as the number of malicious nodes does not exceed one-third of all nodes. PBFT achieves this without relying on

cryptographic assumptions, making it suitable for a wide range of applications. The PBFT protocol operates in a system with a predefined number of nodes, one of which acts as the primary (or leader) node, and the others serve as replica nodes. The primary responsibility of the primary node is to order the requests for service in a sequence and then broadcast them to the replica nodes. The client or user waits for a minimum number of replies from replica nodes (typically a majority) before accepting a response to ensure it is receiving a consistent answer. The protocol operates in a series of rounds, and in each round, the following steps are taken, as shown in Figure 1.4.

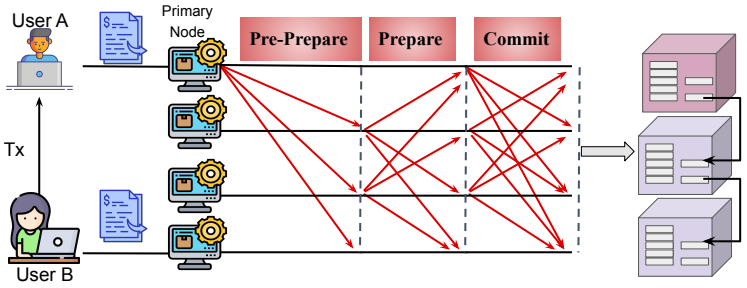


Figure 1.4: Working mechanism of PBFT

1. **Pre-prepare**: As shown in Figure 1.4 this step starts with a leader or primary node, which is periodically changed to ensure fairness and security. This designated leader broadcasts a *pre-prepare* message to the replica nodes in the network, which contains information about the newly proposed block and the current view number. This phase aims to inform the nodes of the new proposal and make a consensus.

2. **Prepare**: Upon receiving the *pre-prepare* message, the replica nodes broadcast a *prepare* message, which indicates that they have received the proposal from the leader node and are ready to execute it. The *prepare* message includes the digest of the proposed block, serving as an endorsement of the proposal's integrity and the leader's legitimacy. To move forward for the next step, i.e., *commit*, there should be positive responses from at least $2/3^{rd}$ of all nodes, i.e., $2f+1$ nodes, where f are the faulty nodes. This step serves as a safeguard against faulty or malicious leaders proposing invalid blocks.

3. **Commit**: Once a node has received 2/3 prepare messages from all nodes (including its own), it broadcasts a *commit* message indicating its readiness to *commit* the proposed block to its local copy of the blockchain. The proposed block is committed to the blockchain if the node receives *commit* messages from $2/3^{rd}$ of all the nodes. This shows that the majority of the nodes agree to the proposed block. The *commit* phase finalizes the consensus process by ensuring that a sufficient number of nodes have agreed to the new block. Once committed, the block is considered final, and the system moves to the next round of consensus process.

Delegated Proof-of-Stake

DPoS is a consensus algorithm designed to enhance the efficiency and scalability of blockchain networks by addressing the limitations of traditional PoW and PoS algorithms. The concept was introduced by Daniel Larimer [11] as part of the BitShares blockchain. DPoS relies on a reputation-based voting system, where token holders vote to elect a small group of delegates (also known as witnesses) to validate transactions

and produce blocks on behalf of the network. This method reduces the number of nodes involved in the consensus process, thus significantly improving transaction throughput and reducing latency. The DPoS protocol operates in a network where the stakeholders (token holders) participate in a voting process to select delegates. These delegates are responsible for maintaining the blockchain, verifying transactions, and generating new blocks. The stakeholders hold the delegates accountable, as they can be voted out at any time if they fail to act in the network's best interest. The DPoS protocol follows a sequence of steps to reach a consensus, which ensures that the blockchain remains secure and efficient. The detail of DPoS is given below.

1. **Election of Delegates**: The first step in the DPoS process is the election of delegates. Token holders use their voting power, proportional to their stake in the network, to elect a group of trusted delegates. These delegates are responsible for block production and transaction validation. The number of elected delegates is typically much smaller than the total number of nodes in the network, which helps in achieving faster consensus.

2. **Block Production**: Once elected, delegates take turns producing blocks in a round-robin manner. Each delegate is given a time slot during which they are responsible for creating a block. This block contains the transactions that have been broadcasted to the network. If a delegate fails to produce a block within their designated time slot, the opportunity passes to the next delegate, and the missing block is left empty, ensuring that the network continues to operate smoothly.

3. **Validation and Approval**: After a block is produced, it is broadcast to the network for validation. The other delegates review the block to ensure it adheres to the network's rules and does not contain any invalid transactions. Once validated, the block is added to the blockchain. The reputation system within DPoS ensures that delegates are motivated to act honestly, as their position depends on maintaining the stakeholders' trust.

4. **Voting and Governance**: In addition to block production, DPoS includes a built-in governance mechanism where stakeholders can vote on protocol changes, system upgrades, or the replacement of underperforming delegates. This democratic process ensures that the network remains adaptable and responsive to the needs of its users.

1.3 SMART CONTRACTS

With the emergence of the Ethereum blockchain, smart contracts came into existence, which are self-executing programs written in Solidity programming language. Smart contracts run on the blockchain and are automatically triggered when certain predefined conditions are met. That is, the execution of smart contracts does not require any intermediaries. The smart contracts programs contained agreements with certain conditions. These contracts exist across a distributed and decentralized blockchain network, which enables the network to execute the trusted transactions among the anonymous participants without the need for central authority [16]. Smart contracts have enabled a wide range of applications, from automated token sales to complex decentralized applications that run on the Ethereum blockchain network. For example, Let user A (buyer) agree to buy a certain service from user B (seller) using a smart contract. The contract is written into the blockchain and might stipulate that when B provides the digital token for the car's ownership, the agreed amount of cryptocurrency from A's wallet is automatically transferred to B's account.

This exchange is transparent, where all parties can verify the transaction, and once executed, it cannot be reversed. While smart contracts offer numerous benefits, they are not without challenges. The host to execute the smart contracts is a blockchain network, which is immutable; therefore, it cannot be modified once the smart contract is executed on the blockchain. Thus, the smart contract vulnerabilities (i.e., bugs or oversights in programming) can lead to unexpected results. The vulnerable contracts lead to unwanted forks in the blockchain. Despite these challenges, there are various applications of smart contracts. One widely known smart contract application is Decentralized Autonomous Organization (DAO) [17]. DAO is governed by smart contracts, making operations independent and transparent. It is a token based governance, where decision making is distributed among the network participants according to their token/stake holding.

1.4 TYPE OF BLOCKCHAIN

Broadly, blockchain is categorized into three types: public blockchain, private blockchain, and consortium blockchain. This categorization is designed with specific use cases and how the nodes are added to the network. Below, we have briefly defined all these categories.

1.4.1 Public Blockchain

A public blockchain is a type of blockchain that allows individuals to participate and opt-out at any moment of the time. It is also called a permissionless blockchain because participating and opting out of the network does not require any permission procedure. The native cryptocurrencies and tokens associated with this blockchain are used as incentives for the participants. The participants can join the network to read, write, and participate in the consensus process. This type of blockchain is fully transparent, decentralized, and secure through a consensus process. Despite these advantages, there are multiple challenges, such as scalability, privacy, and limited performance. The widely known examples include Bitcoin, Ethereum, and Litecoin.

1.4.2 Private Blockchain

Opposite to public blockchain, a private blockchain is a type of blockchain that is operated under a controlled environment. That is, the joining and opting out requires permission; therefore, it is also called a permissioned blockchain. It is useful to organizations that want to use the benefits of blockchain's secure and efficient ledger capabilities but are afraid to put their data in the open network. In contrast to the public blockchain, the private blockchain offers fast transaction confirmation time and higher throughput due to the comparatively simple consensus algorithms that operate amongst trusted and authenticated participants. The customizable nature of these blockchains enables organizations to fine-tune them to fit their requirements. Operating within a controlled environment also means private blockchains can function with less redundancy and resource expenditure than their public blockchains, which makes them cost-effective. The widely known examples include Hyperledger Fabric [18] and Corda [19].

1.4.3 Consortium Blockchain

Consortium blockchains offer a hybrid approach that incorporates the benefits of both public and private blockchains. Selected nodes manage the network of a consortium blockchain, thus providing partial decentralization. This partial decentralization balances the tradeoff between shared control and independent operations. While it allows a higher degree of transparency and security than private blockchains, it restricts participation in the consensus process to a limited number of approved participants. Consortium blockchains are often used in sectors where multiple stakeholders need to securely and transparently share information, such as banking, healthcare, and cross-industry collaborations. A known example of a consortium blockchain is Quorum [20].

In conclusion, the selection amongst the public, private, and consortium blockchains directly depends on the requirement of specific applications and operational models (i.e., open and restricted). A public blockchain is ideal for scenarios demanding transparency and wide participation, and a private blockchain offers an optimal solution for enterprises requiring control and confidentiality. Consortium blockchain is suitable for collaborative projects among multiple stakeholders. In Table 1.1, we provide a comparative analysis between public, private, and consortium blockchains.

Table 1.1: Comparative analysis between Public, Private, and Consortium blockchains

Feature	**Public Blockchain**	**Private Blockchain**	**Consortium Blockchain**
Access	Open to anyone	Restricted, permissioned access	Restricted, permissioned, controlled by multiple organizations
Decentralization	Fully decentralized	Centralized	Partially decentralized
Governance	Community-based or algorithmic	Single organization or consortium	Multiple organizations with shared control
Speed	Slower due to consensus protocols	Faster as there are fewer nodes	Moderately fast, optimized for scale of participants
Security	High, distributed risk	Dependent on the central authority	High, with shared responsibilities
Transparency	Fully transparent	Limited transparency, known participants	Selectively transparent, known participants
Control	No single entity has control	Single organization has control	Multiple pre-selected entities share control
Consensus	PoW, PoS, etc.	Could be simplified consensus	Often more complex than private, less than public
Scalability	Challenging due to size and consensus complexity	Easier due to fewer nodes	Easier than public, more complex than private
Energy Consumption	Potentially high in PoW systems	Lower due to fewer nodes and efficient consensus	Varies, generally more efficient than public blockchains
Use Case Examples	Cryptocurrencies, global computing platforms	Supply chain, internal record-keeping	Inter-bank settlements, consortium led supply chains

1.5 CHALLENGES IN BLOCKCHAIN

Despite many advantages, there are various challenges in blockchain technology that need to be addressed, such as scalability, interoperability, and privacy. Scalability remains a significant challenge in blockchains like Bitcoin, which struggles with limited transaction throughput. Interoperability to enable cross-chain

communication between different blockchain platforms is another challenge. Additionally, the privacy of users and transactions is a big concern, specifically when blockchain is being adopted in applications such as energy trading in smart grid systems. Apart from these challenges, there are some important vulnerabilities in existing blockchain protocols. The vulnerabilities include double-spending attacks, illegal forking, Liveness, 51% attack, and 1% attack. Below, we have defined all these challenges and vulnerabilities in detail.

1.5.1 Scalability

As blockchain systems are decentralized, every transaction must be verified across potentially thousands of nodes. This process ensures security and integrity but limits transaction speed and increases costs [21]. Public blockchains like Bitcoin can handle almost 7 transactions per second, while Ethereum can process about 30. This is much less than traditional payment systems like Visa, which can handle thousands of transactions per second. Layer-2 solutions [22], sharding [23], rollups [24], and Plasma [25] are the techniques to address this issue. Layer-2 solutions, sharding, and sidechains are the prominent techniques to address the scalability issue [26]. From these three techniques, Sharding has emerged as a potential solution to the scalability challenges faced by blockchain networks. However, recent evaluations of state-of-the-art blockchain sharding protocols have revealed significant limitations and challenges, such as securing sharded networks against attacks, robust leader selection mechanisms, and high cross-shard communication overhead. In sharding, the network is divided into small groups called shards. These shards are prone to attacks (i.e., shard takeover or 1% attacks), making the whole blockchain network vulnerable to attackers. These vulnerabilities highlight the need for more secure, robust solutions to prevent attacks. Additionally, an effective leader selection mechanism is the critical component of the efficient performance of the sharding-based blockchain. The state-of-the-art schemes of leader selection in sharded blockchain are either biased or prone to attacks. Therefore, an unbiased and effective leader selection is needed to achieve the highly performing blockchain. Furthermore, high cross-shard communication overhead is another hurdle in achieving the expected outcome of the sharded blockchain. Due to this, the throughput of the blockchain decreases instead of increasing. Inappropriate shard allocation and transaction placements into shards are the root cause of this performance bottleneck.

1.5.2 Interoperability

Interoperability in blockchain systems refers to the ability of different blockchain networks to communicate, share data, and conduct transactions with one another seamlessly, i.e., enabling cross-chain communication. It is a foundational aspect that can significantly determine blockchain technology's widespread adoption and utility. However, interoperability presents various challenges, such as the presence of third-party intermediaries, different blockchain platforms, and transactional finality [27]. The presence of a third party as an intermediary between two blockchains is prone to a direct attack from the adversary, and the trust in the third party is questionable. The existing cross-chain communications are based on the third party intermediary. Designing an interoperable blockchain protocol without a third party intermediary is the biggest challenge. Apart from this, blockchains can vary significantly in their underlying protocols. They may use different consensus mechanisms (e.g., PoW, PoS, and DPoS), have varying block validation processes, and handle state management and storage differently. These variations make it challenging to achieve direct communication and transaction execution across chains. Furthermore, different blockchains have different ways of achieving transaction finality. Some may consider a transaction final after a certain number of block

confirmations (e.g., six blocks in Bitcoin), while others may use alternative mechanisms. Harmonizing these differing finality conditions is necessary for interoperability [28].

1.5.3 Privacy

Public blockchains provide an open ledger where transaction details are visible to all the participants of the network. While this ensures transparency, it also leads to potential privacy concerns. Additionally, blockchain transactions are pseudonymous, not anonymous. Though blockchain transactions do not directly reveal identities, they can often be linked to individuals or organizations through analysis or off-chain information, another cause of privacy breaches. Privacy issues magnify when blockchains are adopted in applications beyond cryptocurrencies, such as in energy trading in smart grid [29], where the exposure of personal data can have far-reaching implications. These systems often handle sensitive personal information that, if not protected properly, could result in privacy violations [30]. There are various possible solutions available to handle privacy concerns; out of them, zero-knowledge proofs (ZKP) are the best-suited solution with some appropriate modifications. The ZKP is a cryptographic principle, which is essential where privacy must be preserved, like in blockchain. ZKP is a cryptographic method where one party can prove to another that a statement is true without revealing any specific details about the statement itself. Cryptocurrencies like Zcash [31] employ ZKPs (specifically zk-SNARKs [32]) to ensure transactions are recorded, but details like sender, receiver, and transaction amount remain concealed [33].

1.5.4 Vulnerabilities in Blockchain

Despite the inherent security advantages, several vulnerability issues threaten the stability and trustworthiness of blockchain networks. Among these, some of the most critical attacks include double-spending, forking, Liveness, 51% Attack, and 1% Attack or Shard takeover attacks. Below, we describe all these vulnerabilities in detail.

1. **Double-Spending Attack:** The double-spending attack is a fundamental problem in transaction processing governed by blockchain technology. It occurs when a user manages to spend the same digital currency more than once. There are multiple ways to launch this attack; for example, an attacker or adversary manipulates the delays to broadcast the conflicting transactions, which get included in two different blocks. Although the consensus algorithms are designed to prevent this attack. However, attackers exploit the vulnerabilities in these algorithms and take advantage of them to include duplicated transactions or invalid transactions in the confirmed block. This attack is usually the resultant attack of other attacks, such as forking and Liveness, which we have defined below. The mathematical foundation for the probability of a double-spending attack succeeding could be derived based on the Poisson distribution.

   - q be the fraction of the total network hash rate controlled by the attacker,
   - p be the fraction of the total network hash rate controlled by honest nodes, with $p+q=1$,
   - z be the number of confirmations the network requires.

   The probability P that the attacker catches up from a deficit is given by:

$$P(z) = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \tag{1.1}$$

where λ is the expected number of blocks the attacker should have found by this point, which is $\lambda = z\frac{q}{p}$. This function describes the cumulative probability that the attacker will find at most z blocks in the period, reflecting the difficulty of the attack as the number of confirmations z increases.

2. **Forking:** Forking in blockchain usually refers to the creation of an alternative chain where two blocks have the same parent block. A fork in the chain is shown in Figure 1.5. There are two types of forks in the blockchain: soft fork and hard fork. A soft fork is a backward-compatible update in the existing blockchain protocol. It requires a majority of the network nodes to update in order to enforce new rules. Segregated Witness (SegWit) is an example of a soft fork in the Bitcoin blockchain [34], [35]. A hard fork is a disruption to the existing blockchain protocol [36]. It is a planned fork, often used to add significant new features. Bitcoin Cash (BCH) [37], [38] is an example of a hard fork in the Bitcoin blockchain, which increased the block size limit from 1 MB to 7 MB. Apart from the soft and hard forks, forking could be legitimate and illegitimate. A legitimate fork is when honest nodes update or create new features in the existing protocol, whereas an illegitimate fork occurs when malicious or unauthorized nodes change the protocol with the intent to disrupt the blockchain protocol. The objective of an illegitimate fork could be to perform a double-spending attack or create a fraudulent version of the blockchain. An illegitimate fork typically occurs when an attacker secretly mines blocks, adds them to the ledger and makes it longer than the honest chain [39].

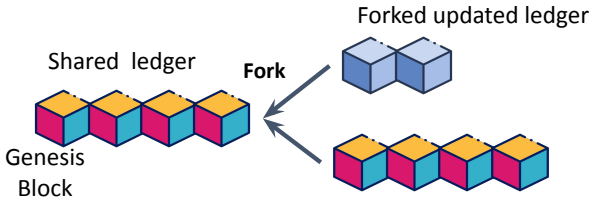


Figure 1.5: Fork in blockchain

3. **Liveness Attack:** Liveness in blockchain refers to the ability of continuous transaction processing and adding new blocks to the ledger. Therefore, a Liveness attack is said to occur when the attacker succeeds in halting the progress of the blockchain. This could happen when the attacker disrupts the network's operation by flooding it with fake transactions or blocks. The Liveness of the blockchain can also be disrupted when the attacker controls a significant portion of the network to prevent the confirmation of legitimate transactions. Let T_{normal} be the transaction confirmation time in normal conditions, and T_{attack} be the transaction confirmation time when the attack occurred. Similarly, B_{normal} and B_{attack} are the block production rates in normal and attack conditions, respectively. A simplified formula to show the impact of the Liveness attack (LA) is represented as follows:

$$LA = \alpha\left(\frac{T_{attack} - T_{normal}}{T_{normal}}\right) + \beta\left(\frac{B_{normal} - B_{attack}}{B_{normal}}\right) \tag{1.2}$$

The α and β are the weighting factors that reflect the relative importance of each metric (i.e., transaction confirmation and block production) to the blockchain's liveness. This formula gives a quantitative measure of the liveness impact, with higher values of LA indicating that the attack impact is high. During the attack, it is expected that the transaction confirmation time increases and the block production time decreases.

4. **1% Attack or Shard Takeover Attack:** The 1% attack is one of the major issues in sharding based blockchain systems. It exploits the reduced security that can occur within individual shards. In

a traditional, non-sharded blockchain, an attacker typically needs to control most of the network's hashing power (in PoW systems) or stake (in PoS systems) to compromise the system, often described as a 51% attack. Meanwhile, in a sharded blockchain, the network's total resources are equally divided among all the shards. In this scenario, controlling a single shard requires significantly less computation than controlling half of the network. For example, if a blockchain network is divided into 100 shards, an attacker might only need to control 1% of the total network to take over a single shard, hence the term "1% attack." Once a shard is compromised, the attacker could make targeted manipulations in the transactions within that shard and violate the integrity of the transactions. This vulnerability highlights the need for carefully designed security measures in sharded blockchains.

1.6 BLOCKCHAIN APPLICATIONS

In this chapter, we have discussed various aspects of blockchain technology, from its fundamental background to the inherent challenges it faces. However, this discussion would be incomplete without exploring the practical applications of this technology. Blockchain gained prominence primarily due to the success of cryptocurrencies, especially Bitcoin. As a result, blockchain is widely recognized for its role in the cryptocurrency market. However, researchers have identified its potential in many other fields, such as healthcare, smart cities, transportation, energy distribution, and more. This thesis is dedicated to three applications: the Internet of Things (IoT), Intelligent Transportation Systems (ITS), and Smart Grids. These applications are critical components of any future smart city. However, integrating blockchain technology with these applications is not as simple as using plug-and-play tools due to the existing challenges and issues associated with blockchain technology, as discussed earlier. In the envisioned smart cities of the future, blockchain's role will be pivotal in ensuring data security, integrity, and decentralization. As urban infrastructures become increasingly interconnected through IoT devices [40], [41], the enormous amount of data generated necessitates a secure and tamper-resistant storage mechanism. Blockchain's decentralized structure emerges as an ideal solution, ensuring that diverse data streams—spanning traffic control and energy distribution—are preserved with integrity and transparency [7]. Therefore, an enhanced and improved blockchain version must be investigated before integrating this technology into applications. Below, we outline the three applications for which we have proposed blockchain-based security solutions.

1.6.1 Internet of Things (IoT)

The transformative potential of IoT promises both substantial economic benefits and societal advancements [42]. It envisions an interconnected world, facilitating seamless communication and service delivery. Such advancements have been instrumental in the evolution of smart cities, especially in areas like smart homes, smart healthcare, and environmental monitoring [43]. However, the security of these interconnected systems is highly essential. Conventional security measures often fall short in IoT applications due to the high computational requirements needed to maintain data security. Moreover, IoT devices' limited computational power and energy constraints mean that traditional cryptographic methods are often impractical. Furthermore, conventional IoT application security solutions are based on a centralized architecture, which is prone to a single point of failure. Therefore, we have designed three schemes for blockchain-based security solutions in IoT applications, namely, Secure Communication and Authentication for IoT Applications using Blockchain (SCAB-IoTA), Blockchain-based Integrated Security System for Internet of Things (IoT) Applications (BLISS), and Blockchain-Enabled Security System for Internet of Things Applications (BSS).

1.6.2 Intelligent Transportation Systems (ITS)

Modern smart cities face significant challenges in traffic management in transportation systems. The statistics of global and Indian road accidents emphasize the design of efficient ITS and show why it is one of the biggest challenges in traffic management. For instance, road accidents rank as the ninth leading cause of death globally, claiming approximately 1.5 million lives annually [44]–[46]. Efficient ITS offers a solution that aims to enhance road traffic efficiency [47]. With the inclusion of IoT devices and interconnectivity, the performance of the ITS increases many folds. However, this inclusion exposes the system to potential vulnerabilities to the attacker. Blockchain offers a solution to address these vulnerabilities. Therefore, we have designed two solutions for ITS based on blockchain technology, which are a Lightweight Blockchain Security Protocol for Secure Storage and Communication in SDN-Enabled IoV (LBSV) and A Novel Incentive Mechanism for Vehicles in Intelligent Transportation System Based on Consortium Blockchain (SmartCoin).

1.6.3 Smart Grid

With the emergence of smart cities and blockchain technology, the demand for energy is also increasing. Conventional energy sources are either highly polluting, highly costly, or both. The green energy is the better alternative [48]. However, the distribution of this energy is challenging when it comes to the individual households generating it on the rooftop of the house [49]. Modern smart grids, strengthened with internet capabilities, promise dynamic energy distribution and pricing. Yet, their centralized architecture is prone to data compromises and systemic failures. Blockchain offers promising solutions to these issues. While blockchain holds the potential to secure smart grid energy trades, succeeding protocols are restricted by computational constraints and inefficiencies. Existing cryptocurrencies also struggle to transparently facilitate energy trading, with performance and security issues [29], [50]–[52]. Therefore, to address these challenges, we have designed three schemes for energy trading, which are blockchain Protocol for Secure Transaction Processing in the Next Generation Smart Grid (SmartGrid-NG), Blockchain-Based Energy Trading in the Local Energy Market (LEM) Using Modified Double Auction Protocol (ETradeChain), and Privacy-Preserved Blockchain Framework for Energy Trading in Smart Grid Using Zero Knowledge Proof (zk-SGB).

1.7 MOTIVATION AND CONTRIBUTIONS

Having established a foundational understanding of blockchain technology, its consensus mechanisms, and the diverse applications and challenges inherent in its integration across various domains, this thesis is driven by the need to address several pressing issues that hinder blockchain's broader adoption and efficacy. While blockchain offers unprecedented opportunities for decentralization, security, and transparency, significant challenges remain in areas such as scalability, interoperability, and security, particularly in complex and dynamic environments like IoT, ITS, and smart grids. This thesis aims to tackle these challenges by posing critical research questions and developing novel solutions that push the boundaries of what blockchain technology can achieve. Each chapter is dedicated to exploring these questions in depth, offering innovative contributions that advance the state-of-the-art in blockchain research and its practical applications.

The scalability of blockchain has been explored through questions such as: "*How can blockchain systems be designed to handle a growing volume of transactions without compromising decentralization and security?*", "*What novel consensus mechanisms or structural modifications can be introduced to overcome the inherent limitations of existing sharding-based solutions?*," and "*How can cross-shard communication and synchronization issues be mitigated to ensure high throughput and low latency in scalable blockchain systems?*" These questions are addressed in Chapter 3 of the thesis. Interoperability in blockchain has been investigated with questions like: "*How can different blockchain platforms be made to communicate and transact with each other without relying on third-party intermediaries?*", "*What algorithms or protocols can be developed to enable secure, trustless, and efficient cross-chain asset transfers?*," and "*How can we ensure that the fundamental characteristics of blockchain (non-repudiation, unlinkability, confidentiality, and atomicity) are preserved in an interoperable environment?*" Chapter 4 is dedicated to answering these questions with respect to the proposed scheme called CrossLedger.

Additionally, the thesis aims to strengthen the blockchain ecosystem, particularly in IoT, ITS, and smart grid applications, by exploring the following questions: "*What blockchain-based solutions can be implemented to enhance the security of IoT applications, particularly in terms of secure communication and authentication?*", "*How can ITS benefit from blockchain technology in terms of secure data dissemination and protection against common cyber-attacks like Sybil attacks, man-in-the-middle attacks, and data tampering?*", "*What are the trade-offs in integrating blockchain with smart grid systems, and how can we design a secure and efficient energy trading system that addresses these trade-offs?*," and "*How can a new cryptocurrency, tailored for energy trading in decentralized smart grids, be designed to prevent issues like unfair pricing and ensure secure transactions?*" Chapter 5 of the thesis answers all these questions for each application separately. In addressing these research questions about scalability, interoperability, and the blockchain ecosystem, the following are the major contributions of this thesis:

- **Restructured Blockchain Architecture:** The thesis introduces *BlockTree*, a novel blockchain architecture that restructures the traditional linear blockchain into a nonlinear structure, enabling parallel processing of multiple transactions. This restructured design significantly enhances scalability by allowing the blockchain to handle a higher volume of transactions without compromising decentralization or security.
- **Cross-Shard Consensus Mechanism:** The thesis presents *UnoShard*, a sharding-based solution that introduces a novel cross-shard consensus mechanism. This mechanism ensures seamless communication and synchronization between shards, reducing latency and improving throughput in scalable blockchain systems. Additionally, *UnoShard* incorporates a trust-based PBFT consensus mechanism within each shard, enhancing security and reliability in transaction validation.
- **Generalized Cross-Chain Asset Transfer Protocol:** The thesis introduces *CrossLedger*, a pioneering protocol that facilitates cross-chain asset transfers between heterogeneous blockchain platforms without relying on third-party intermediaries. *CrossLedger* employs innovative algorithms such as the Asset Forwarder Selection (AFS), Trust Establishment, and Asset Transfer and Confirmation (ATC) algorithms to ensure secure, trustless, and efficient interoperability between different blockchain networks.
- **Integrated Security System for IoT Applications:** To address the research questions, we proposed two blockchain-based schemes, *BLISS* and *SCAB-IoTA*, to enhance the security of IoT applications. *BLISS* utilizes smart contracts on the Ethereum blockchain to create secure clusters of IoT devices,

providing decentralized authentication and secure data transmission within these clusters. *SCAB-IoTA* incorporating angular distance measurement for clustering movable devices and using a hybrid cryptosystem (ECC and AES) for secure communication. Together, these schemes ensure robust security for IoT devices, particularly in dynamic and mobile environments, reducing risks such as device impersonation and data breaches.

- **Lightweight Blockchain Security Protocol and Incentive Mechanism for ITS:** The thesis introduces *LBSV* and *SmartCoin*, two innovative schemes for enhancing security and efficiency in ITS applications. *LBSV* is a lightweight blockchain security protocol that integrates a modified PBFT consensus algorithm and a hybrid SDN architecture, ensuring secure storage and communication in SDN-enabled IoV networks. *SmartCoin* complements *LBSV* by providing a novel incentive mechanism based on consortium blockchain, which promotes honest behavior among vehicles in ITS through the use of a new digital currency, SmartCoins (STC). Together, these schemes improve secure data dissemination, protect against cyber-attacks, and enhance the reliability and efficiency of the ITS environment.
- **Secure and Efficient Energy Trading in Smart Grids:** This thesis presents three blockchain-based systems namely, *SmartGrid-NG*, *ETradeChain*, and *zk-SGB*, which are designed to provide security, privacy-preservation, and optimized energy trading in decentralized smart grids. *SmartGrid-NG* integrates smart grid technology with the GridChain consensus algorithm, enhancing the security and reliability of energy transactions by eliminating vulnerabilities such as single points of failure. *ETradeChain* introduces a new energy cryptocurrency called Pcoin and employs a modified double auction and Vickrey auction methods to ensure fair and transparent energy trading. *zk-SGB* further ensured the privacy of the user in energy trading. These schemes address key challenges in smart grids, such as transaction security, unfair pricing, and efficient energy distribution, making them crucial for the development of next-generation energy systems.

1.8 THESIS STRUCTURE

Beginning with an introduction to define challenges and vulnerabilities in blockchain in **Chapter 1**, we have covered the related work in **Chapter 2**, providing context for the proposed solutions. **Chapter 3** describes the challenges and solutions to the scalability of blockchain. The proposed schemes BlockTree and UnoShard are described in this chapter to show the potential solution to improve the scalability. The BlockTree is a scalable blockchain solution with a hierarchical structure that significantly improves transaction finality, while the UnoShard is a scheme to harmonize scalability and security in blockchain sharding. **Chapter 4** is dedicated to the interoperability in the blockchain. The proposed CrossLedger scheme addresses the critical need for seamless asset transfers across different blockchain networks, marking a significant step towards blockchain interoperability. The thesis then progresses to practical applications of blockchain in IoT, ITS, and Smart Grids in **Chapter 5**. In this chapter, we first see the schemes proposed for the IoT application, which are BLISS and SCAB-IoTA. After that, we went deeper into the ITS applications and explained the two proposed schemes, LBSV and SmartCoin. After covering IoT and ITS, we explain the blockchain-based solutions for energy trading in the smart grid, which are SmartGrid-NG, ETradeChain, and zk-SGB.

2

Related Work

Consensus algorithms are the core concept of blockchain technology. They facilitate the process of reaching an agreement on specific decisions or actions among the nodes in a blockchain network. This process ensures that the majority of nodes in the network validate and commit to the authenticity of transactions, making consensus algorithms vital for preserving the security and integrity of the blockchain ledger [53]. Various consensus algorithms, such as PoW, PoS, PBFT, and DPoS, are employed by different blockchain platforms. Broadly, these algorithms are categorized into two groups: Proof or resource-based consensus algorithms and Vote or authority-based consensus algorithms [13]. The state-of-the-art schemes either introduce novel consensus algorithms or present modified versions of existing ones. However, the primary objectives of these schemes are to enhance scalability, improve interoperability, or demonstrate the practical applications of blockchain in various domains such as IoT, ITS, and Smart Grids. Consequently, we categorize the related work into four areas: consensus enhancements, scalable blockchain schemes, interoperable blockchain schemes, and blockchain applications. The subsequent sections provide an in-depth exploration of these state-of-the-art schemes.

2.1 CONSENSUS ALGORITHM ENHANCEMENTS

There are many researchers who proposed a significant enhancement in the existing consensus algorithms, specifically the PoW, PoS, and the PBFT. Below are state-of-the-art schemes of blockchain enhancements. Peercoin [14] and BlackCoin [54] are the techniques that use the PoS for the consensus amongst the nodes. However, Peercoin is a peer-to-peer cryptocurrency utilizing both PoW and PoS systems. Some PoS cryptocurrencies require a digital signature to sign the block to mitigate the fork digitally. Fitzi et al. [55] proposed a formal execution model for PoS, which enables the expression of the near-optimal transactional throughput. The proposed modified version of the PoS protocol achieves security and optimal throughput against an active adversary who controls the stakes in the system. Kiayias et al. [56] improves the fundamental security threshold of PoS by following the longest chain rule and in the presence of honest and adversarial leaders. Bagaria et al. [57] proposed PoS longest chain protocols to ensure security against nothing-at-stake attacks and use the s-truncated longest chain rule to prevent attacks. However, there is a probability that adversaries could win all leader elections. Karakostas et al. [58] proposed a scheme to secure PoW against majority attacks using checkpointing to reduce the energy and computational resources required for mining. It leads to governance issues due to external entities for checkpointing. Kohlweiss et al. [59] proposed a anonymous PoS. It leverages network delays to distinguish stakeholders. It also

provides liveness guarantees under adversary-controlled network delays, enabling an adversary to determine a node's stake through message delays. However, it is not scalable in a permissionless blockchain setting because of the challenge of finding honest parties distributed across the network. The linear nature of blockchain ensures data security but restricts transaction processing speed. Therefore, Song et al. [60] presents a Layer-based directed acyclic (L-DAG) graph blockchain that maintains block order and employs a modified PBFT protocol. Using the proportional integral derivative controller, the L-DAG ensures ledger consistency. Implemented through Hyperledger Fabric, the L-DAG-based PBFT demonstrates near-linear efficiency in transaction processing as consensus threads increase. L. Daniel proposed a DPoS [11] scheme that relies upon a group of delegates to validate blocks on behalf of all nodes in the network. Coin holders use their coin balances to elect delegates, called witnesses. These witnesses (delegates) create blocks of new transactions and add them to the blockchain. However, it is not good for large networks because there are generally between 21–100 elected delegates in a DPoS system. A leaderless consensus [61] is proposed to avoid leader bottleneck and reduce the impact of a single point of failure on performance. Liu et al. [62] proposed an improved PBFT consensus called Credit (CBFT) enhanced with a grouping and credit model. It shows the low throughput as it is just 3.1 times that of PBFT, and latency is 7.4% that of PBFT. Xu et al. [63] describes a system that tolerates up to half of the nodes being adversaries, improving throughput with parallel shards. Many researchers used two different consensus algorithms for their solutions. Kogias et al. [64] proposed a PoW and PBFT-based consensus protocol called ByzCoin. It is an extension of Bitcoin-NG. A group of leaders generates key blocks instead of one leader. The achieved throughput is about 600-700 TPS with 1008 nodes, which is very low. Additionally, poor scalability and fork generation are the big concerns. A trusted finance supply chain data is proposed in [65]. It is suitable for public environments and mitigates financial risks by ensuring transparent and secure financial transactions.

2.2 STATE-OF-THE-ART SCHEMES FOR SCALABLE BLOCKCHAIN

Bitcoin is the first broadly adopted cryptocurrency that uses the PoW [1] consensus algorithm to agree on the validation of the transactions. In Bitcoin, PoW takes 10 minutes to add every next block into the blockchain ledger [66], [67], and it takes approximately one hour or at least six blocks to confirm the transaction [68], [69]. Hence, the throughput is very low. Bitcoin-NG [70] solves this issue of Bitcoin and proposes a new protocol that uses the PoW for leader selection. It uses two types of blocks, key block and microblock, and the operation is organized into periods called epochs. The distinction between a microblock and a key block is that the key block does not contain any transaction, but the microblock does. The key issues of Bitcoin-NG are the unfair selection of leaders, the remuneration policy, and the higher possibility of forks. The ByzCoin [64], which is based on the PBFT protocol, extends the Bitcoin-NG [70] to achieve high transaction throughput. The key block generation has been modified, where the group of leaders generates the key block instead of a single leader. The remuneration model also changed where the incentives are distributed within the peer nodes in proportion to their stakes. The leader runs a modified version of the PBFT to generate a combined signature that confirms at least two-thirds of valid responses. A node can verify in constant time that the consensus group has validated a macroblock. A milestone approach called Algorand [71] is a byzantine agreement protocol that uses Verifiable Random Functions (VRF) for the selection of nodes for consensus. Algorand's achieved throughput is 125 times that of Bitcoin. Obscuro [72] provides anonymity to Bitcoin transactions. The trusted execution environment gives high efficiency to the protocol. Conflux [73] enhanced the performance with high throughput and fast confirmation time using a scalable solution. Conflux arranged the blocks in a Tree-Graph structure. In Tree-Graph, the blocks

Table 2.1: Comparison of consensus algorithm enhancements

Scheme	Consensus Algorithm	Advantages	Limitations
Peercoin [14]	PoS	First peer-to-peer cryptocurrency based on PoS reduces the energy consumption of PoW by limiting the hashing search space	1. Potential misuse of the coin age concept. 2. Potential centralization due to centralized checkpointing
BlackCoin [54]	PoS	It is PoS version-2 that resolves the issues of original PoS, which is adversarial use of coin age	Potential susceptibility to attacks due to lower proof computation
Fitzi et al. [55]	PoS	Near-optimal transactional throughput against active adversaries	Adversarial stake control pose risks
Kiayias et al. [56]	PoS	Improved security threshold by honest/adversarial leader distinction	Adversarial control during leader elections
Karakostas et al. [58]	PoW + Checkpointing	Reduces energy/resource usage, enhances security	Governance issues due to external entities, potential centralization
Kohlweiss et al. [59]	PoS + Anonymity	Used the PoS with anonymity and ensures liveness under network delays and privacy preservation	Not scalable in permissionless settings and network delay reveals stakes
Song et al. [60]	PBFT + L-DAG	Used the PBFT consensus along with layered based DAG structure and provides near-linear transaction processing	Complex implementation, limited to Hyperledger environments
L. Daniel [11]	DPoS	Relies upon a group of delegates to validate blocks on behalf of all nodes in the network. Coin holders use their coin balances to elect delegates, called witnesses. These witnesses (delegates) create blocks of new transactions and add them to the blockchain.	Not good for large networks because there are generally between 21–100 elected delegates in a DPoS system. Fast compared to PoS but partially centralized.
Antoniadis et al. [61]	Leaderless Consensus	Used leaderless consensus to avoid leader bottleneck and enhances fault tolerance	Potentially higher communication overhead, complexity in implementation
Liu et al. [62]	PBFT + Credit (CBFT)	This also used PoW and PBFT together for transparent and secure financial transactions	Architectural design leads to potential centralization and high consensus delay
Xu et al. [63]	PoW + Parallel Sharding	Tolerates up to half nodes as adversaries, improved throughput	Complexity in sharding management, cross-shard communication issues
Kogias et al. [64]	PoW + PBFT (ByzCoin)	Used the PoW and PBFT in conjunction to improve throughput and distributed incentive model	Issue of forking and potential double-spending
Liu et al. [65]	PoW + PBFT	Transparent and secure financial transactions	High consensus delay, potential centralization

are linked based on the happened-before relationship; thus, concurrent blocks are not considered conflicts. The block confirmation time is about one minute, which can significantly reduce to achieve high throughput. Another protocol called OmniLedger [74], a sharding-based distributed ledger scheme, was proposed, which introduces a cross-shard commit protocol for intra-shard transaction processing.

Monoxide [75] runs multiple independent and parallel instances of single-chain consensus systems termed zones. However, these zones are susceptible to double spending and forking attacks. HoneyBadger [76] proposed by Miller et al. assigns a collection of servers to reach consensus for validated transactions. The achieved throughput is 1500 transactions per second with 104 nodes. The scheme also shows the high consensus delay and partial centralization. A scheme called RedBelly [77] proposed by Crain et al. claims to be fork-proof. A lack of decentralization and trust is identified due to the restricted participation of pre-approved nodes. RapidChain emphasizes fully sharded solutions [78], while ELASTICO offers a secure sharding protocol for open blockchains [79]. Chainspace focuses on transaction processing with a unique node structure [80], and research stresses the importance of shard-persistence and shard-liveness for maintaining safety and liveness in transactions [23]. PBFT's frequent inter-node communications hinder its scalability, making it suitable for smaller networks. To address this, Li et al. [81] introduce a scalable multi-layer PBFT mechanism that hierarchically groups nodes into different layers, restricting communication within these groups. An optimal double-layer PBFT is proposed, which minimizes communication complexity when nodes are evenly distributed within the second layer's sub-groups. Marson et al. [82] proposed a novel schemes for scaling blockchain called Mitosis that enhances throughput by dynamically splitting and merging blockchain shards based on network demand. Mitosis achieves high scalability and maintains security through a combination of parallel processing and a consensus protocol designed for sharded blockchains, making it a robust solution for large-scale distributed networks. The major drawback of Mitosis is the complexity of managing shard splits and merges, which can lead to increased latency and potential inconsistencies during high-traffic periods. Gazi et al. [83] proposed a blockchain transaction settlements estimation that facilitates trust in decentralized systems, which supports scalability. However, it is resource-intensive. Cai et al. [84] proposed a scheme called Benzene for scaling blockchain With a cooperation-based sharding design for a cross-shard block verification mechanism leveraging a Trusted Execution Environment (TEE). Yu et al. [85] proposed a sharding approach called Overshard for scaling blockchain that incorporates an overlapping network. The overlapping network model enables all nodes in a shard to join multiple other shards. Due to the incorporation of overlapping networks, the communication overhead is higher. Singh et al. [86] proposed a blockchain-based decentralized trust management scheme using smart contracts. Specifically, the author introduces the concept of blockchain sharding to reduce the load on the main blockchain and increase the transaction throughput. However, the high sharding overhead poses a significant hurdle in achieving high performance. Bagaria et al. [87] proposed a novel blockchain protocol called PRISM that decouples traditional blockchain architecture into proposer, transaction, and vote blocks. This multi-block approach enables high transaction throughput and fault tolerance, significantly improving scalability and efficiency. However, the added complexity and resource intensity pose challenges in implementation and maintenance. The PRISM authors further improve the PRISM protocol's blockchain performance in [88] and [89]. They aim to scale Bitcoin by a factor of 10,000. The protocol introduces optimizations for proposer block generation, parallel processing of transaction blocks, and an enhanced vote block mechanism, achieving unprecedented scalability and robustness. Nevertheless, this scalability comes at the cost of increased network overhead and potential synchronization issues. A common bottleneck of the above schemes is that they face a high cross-shard communication overhead in sharding blockchain. Additionally, the biased leader selection is prone to many vulnerabilities. Consequently, these state-of-the-art schemes face the performance issue, i.e., low throughput. Some of the schemes improved the performance of blockchain using the sharding concept. However, incorporating sharding brings new

challenges to the blockchain, such as cross-shard communication overhead and shard takeover attacks. As a result, the performance has not improved as expected. The proposed solutions BlockTree and UnoShard address these challenges in the sharding-based blockchain.

2.3 STATE-OF-THE-ART SCHEMES FOR INTEROPERABLE BLOCKCHAIN

Blockchain technology's rapid evolution has extended its impact beyond cryptocurrencies, creating a growing need for interoperability among the increasing number of blockchain platforms. Interoperability is crucial for enabling seamless asset transfers, data sharing, and communication across different blockchain ecosystems. The survey papers [27], [28] provide a comprehensive overview of the current state of blockchain interoperability, which states that interoperability approaches can be categorized into various types, including sidechains, notary schemes, hash-locking, and blockchain of blockchains. They also highlight key challenges, such as standardization, security, and the presence of a third party as an intermediary, which must be addressed to achieve effective interoperability. Developing universal standards and more robust protocols is essential for overcoming these limitations and advancing blockchain interoperability. In this section, we have provided detailed and comprehensive work related to state-of-the-art schemes that addressed various issues of blockchain interoperability. Zamyatin et al. [90] provides a systematic overview of communication protocols between distributed ledgers focusing on achieving secure and efficient interoperability. The study concludes that while significant progress has been made, there is still a need for more robust and scalable communication protocols to support widespread blockchain interoperability. A novel platform for smart contracts and decentralized apps for enabling interoperability across multiple blockchains has been done by the makers of Ethereum in [6]. Kwon et al. [91] designed a system called Cosmos that aims to address the challenges of blockchain scalability and interoperability by enabling cross-chain communication while retaining the security properties. However, its reliance on the Cosmos Hub as a central point of failure could pose risks. Gazi et al. [83] addressed the challenge of determining settlement times in blockchain systems, which is critical for ensuring secure and reliable cross-chain transactions. However, this approach is limited by the inherent delay and energy inefficiencies associated with PoW mechanisms, which hinders its applicability in high-throughput interoperability scenarios. Jin et al. [92] proposed an architectural framework designed to facilitate interoperability and emphasize modularity, allowing different blockchain systems to integrate through standardized interfaces. While the architecture offers a flexible solution to interoperability, it is still in the conceptual stage and lacks a practical implementation that can demonstrate its effectiveness in real-world scenarios. Pillai et al. [93] introduced a protocol called Burn-to-Claim, a novel approach to asset transfer across multiple blockchains by requiring assets to be burned on the source blockchain before being claimed on the destination blockchain. Burn-to-Claim addressed the double spending challenges by ensuring that assets are not duplicated across chains. However, the protocol's reliance on asset burning can be seen as a limitation, as it permanently removes assets from the source blockchain, which may not be suitable for all use cases. Su et al. [94] accomplish the task of cross-chain exchange using transaction dependencies and conditional transactions. The approach allows for the secure exchange of assets across blockchains without the need for a trusted third party. However, this scheme is very complex to implement and requires all participating blockchains to support conditional transactions, which limits its applicability. Sigwart et al. [95] used the combination of atomic swaps and smart contracts to design a fully decentralized protocol for asset transfer between blockchains without the need for intermediaries. Although this approach enhances security and decentralization, it suffers from scalability issues, particularly when dealing with a large number of transactions or blockchains. Besanccon et al. [96] proposed a framework

Table 2.2: Comparison of state-of-the-art schemes for scalable blockchain

Scheme	Consensus Algorithm	Description	Limitations
Bitcoin-NG [70]	PoW	Decouples blockchain operation into Leader selection and transaction serialization Two types of blocks, keyblock and microblock	Unfair leader selection and remuneration policy, potential fork generation, and low throughput up to 20 TPS.
ByzCoin [64]	PoW + PBFT	An extension of Bitcoin-NG Group of leader generate key block instead of one leader. Remuneration is distributed based on stakes	Poor Scalability, fork generation, low throughput 600-700 TPS with 1008 nodes
Algorand [71]	VRF + Byzantine Agreement	Byzantine agreement protocol called BA*. Uses Verifiable Random functions for the selection of nodes for consensus	Dependency on VRF security and low throughput as of 125 times Bitcoin's throughput
Conflux [73]	PoW + Tree-Graph	Conflux operates with a novel consensus protocol which optimistically processes concurrent blocks without discarding any as forks and adaptively assigns weights to blocks based on their topologies in the Conflux ledger structure (called Tree-Graph)	Complex structure and vulnerability to double-spending attacks
OmniLedger [74]	ByzCoinX	Sharding based distributed ledger scheme. Introduce cross-shard commit protocol Parallel intra-shard transaction processing	Potential shard takeover attacks, fork generation, low scalability
Monoxide [75]	Asynchronous Consensus Zones	Running multiple independent and parallel instances of single chain consensus systems termed as zones	Double spending attack due to independent consensus zones and low scalability
HoneyBadger [76]	Asynchronous BFT	Assigns a collection of servers to reach consensus for validated transactions	Partial centralization, High consensus delay, 1500 TPS with 104 nodes
RedBelly [77]	PBFT	Fork-proof, high security	Centralization risks, restricted participation
RapidChain [78]	Optimal intra-committee consensus	Fully sharded solution, high scalability	Cross-shard communication overhead, shard takeover risks
ELASTICO [79]	Classical byzantine consensus	Secure sharding, suitable for open blockchains	High overhead in shard communication, potential for attacks
Chainspace [80]	Sharded Byzantine Atomic Commit (S-BAC)	Node structure optimized for transaction processing	Complexity in shard-persistence and shard-liveness management
PRISM [87]	Proposer-Transaction-Vote Blocks	Prism is based on a structured DAG created by cryptographic sortition of the mined blocks. Divided the process into three steps: proposer block, voter block, and transaction block	Increased network overhead and potential synchronization issues. Added complexity and resource intensity pose challenges in the implementation

for secure data exchange across gaming platforms, allowing players to transfer assets between different gaming environments. However, its focus on a niche area limits its broader applicability. In [97], a pegged sidechains concept was proposed as a more general solution for blockchain interoperability, enabling asset transfers between blockchains without altering the underlying protocols. While promising, this approach presents challenges in maintaining the security and stability of the sidechains. Zarick et al. [98] proposed a novel protocol called LayerZero, which is a trustless omnichannel interoperability protocol. This protocol achieves direct cross-chain transactions without replying to a trusted third party. However, while the approach is innovative, it introduces potential challenges related to the reliance on off-chain entities, which, though independent, could become points of failure or attack in certain scenarios. Xie et al. [99] proposed a novel approach called zkBridge using zk-SNARK. The zkBridge is scalable and privacy-preserving, making it suitable for use in scenarios where data confidentiality is crucial. However, the scheme is computationally inefficient as it shows a high computational overhead. Liu et al. [100] proposed an interoperability platform called HyperService specifically designed for Daaps. The platform used a two-layer architecture consisting of a control layer that manages cross-chain transactions and a data layer that handles the execution of smart contracts. The scheme is highly susceptible to a single point of failure due to the centralized control layer. Despite significant advancements proposed by state-of-the-art schemes for blockchain interoperability, they often fall short in many key areas, such as these solutions being either confined to specific ecosystems or suffering from performance bottlenecks, which limits their applicability in diverse blockchain environments. In contrast, the proposed *CrossLedger* is a comprehensive interoperability solution that overcomes the limitations of these state-of-the-art schemes. *CrossLedger* integrates advanced security features such as atomicity, non-repudiation, unlinkability, and confidentiality, ensuring secure and private cross-chain transactions. Unlike other solutions, it offers true interoperability across various blockchain platforms without requiring significant modifications to existing protocols, providing greater flexibility.

2.4 STATE-OF-THE-ART SCHEMES FOR BLOCKCHAIN APPLICATION

Blockchain technology can be adopted in various applications, including IoT, ITS, and smart grids. Numerous schemes have been proposed in these areas to enhance the performance and security of these applications. In this related work section, we explore several such schemes, discussing their merits and demerits. Hammi et al. [101] proposed a decentralized authentication mechanism called Bubble of Trust, creating secure virtual zones for the IoT application. The high computation overhead, storage overhead, and low throughput are the big concerns of this scheme. Dorri et al. [102] proposed a lightweight and scalable blockchain that forms an overlay network of IoT devices. Another application for IoT is proposed called IoT passport [103], which is a decentralized trust framework for cross-platform collaboration motivated with real, real-life passport system rules written in smart contracts. The low scalability and high computational overhead are the issues of this scheme. Li et al. [104] proposed a blockchain-based data storage and protection scheme. It used the distributed hash table to store the data and blockchain to handle the device's authentication. The computation and storage overhead is very high for lightweight IoT devices. Mirsky et al. [105] proposed anomaly detection using blockchain to ensemble an anomaly detection model for distributed trust. The extensible markov model is used for anomaly detection training. High computation overhead, high consensus delay, and poor scalability are the main concerns of this scheme. Yang et al. [106] proposed a decentralized trust management system in vehicular networks based on blockchain techniques, i.e., PoW and PoS. The Bayesian inference model is used to validate the messages received by the neighboring vehicles. Lau et al. [107] proposed the blockchain-based authentication scheme in the IoT

Table 2.3: Comparison of state-of-the-art schemes for interoperable blockchain

Scheme	Consensus Algorithm	Description	Limitations
Zamyatin et al. [90]	Various (Communication protocols)	Cross-chain communication focusing on achieving secure and efficient interoperability	Limited robustness and scalability
Cosmos [91]	Tendermint	Addressed the challenge of scalability and interoperability by enabling cross-chain communication while retaining security	Potential risk of a single point of failure due to reliance on central Cosmos Hub
Burn-to-Claim [93]	Asset burning mechanism	Introduced protocol called Burn-to-claim, to asset transfer across multiple blockchains and prevents double spending, ensure asset uniqueness	Permanent asset removal from source blockchain results to limited use cases
LayerZero [98]	Trustless omnichannel protocol	Proposed a protocol called LayerZero, a trustless omnichannel interoperability protocol, enables direct transfer with trusted third party	Potential challenge on reliance on off-chain entities, which could become the point of failure and attract attackers
zkBridge [99]	zk-SNARK	Proposed a protocol zkBridge using cryptographic tool zk-SNARK, is a scalable and privacy-preserving suitable for scenarios where data confidentiality is crucial	Computationally inefficient due to high computational overhead.
HyperService [100]	Two-layer architecture	Proposed interoperability platform called HyperService specially designed for Dapps, which used the two-layer architecture	Single point of failure due to centralized control layer

network to authenticate IoT devices when they join the IoT network. The authenticated device configuration protocol (ADCP) is proposed for authentication. However, there is no mechanism to check the integrity of the information, and the scheme is prone to Sybil attacks. Michelin et al. [108] proposed a scheme called SpeedyChain for smart cities to secure the data. They have used the permissioned blockchain. The proposed scheme mainly focused on ITS's security, which is unsuitable for a heterogeneous environment. Cui et al. [43] proposed a hybrid blockchain-based identity authentication scheme for IoT networks. Different types of nodes in the network form the hybrid blockchain network, say local blockchain and public blockchain. However, the scheme is not fully decentralized and, hence, prone to single-point failure. The message size is also very high, approximately 30-350MB, which is huge for the IoT network. Gao et al. [109] proposed a scheme for vehicular networks using blockchain and software-defined networks to enhance network performance and establish trust among peer nodes. Zhang et al. [110] proposed a secure data sharing scheme for the IoV network. The scheme encourages honest participants to give incentives in the form of some cryptocurrency. However, the scheme used the central trusted management authority, which again violates the concept of blockchain, i.e., distributed system, and is highly prone to single-point failure. Additionally, the system's throughput depends on the underlying cryptocurrency, which will be very low if Bitcoin and Ethereum are used. The proposed algorithm used PoW for the leader selection and the PBFT for the consensus on the transactions' finality. Li et al. [111] proposed an incentive announcement scheme for vehicular networks based on blockchain technology called CreditCoin. There is a central, trusted authority that can

be compromised. Trust-based blockchain technology, along with deep learning techniques [112], has been used by researchers to improve the consensus mechanism in software-defined vehicular networks. Huang X. et al. [113] proposed a blockchain-based security model for electric vehicle and charging pile management, i.e., the security of trading between the electric vehicle and charging pile. Kudva S. et al. [114] proposed a secure and practical consensus algorithm called proof of driving (PoD) to address the efficiency, fairness, and scalability issues of PoW, PoS, and PBFT consensus algorithms for blockchain-based VANET systems. The PoD was designed to randomize the selection of honest miners for block generation. The author also introduced a filtering technique based on the service standard score of the vehicular miner nodes to detect and eliminate malicious nodes. An Efficient privacy-preserving authentication model has been proposed in [115] to tackle the problem of high latency membership verification for users in VANET for confidential identity requirements. It is a three-phase process where the first phase is initialization, the second phase is chain construction, and the third phase is authentication. Similarly, in [116], a privacy-preserving vehicular communication is proposed in a combination of consortium blockchain and edge computing to support efficient computation and storage capabilities with low communication delay. To achieve privacy preservation, a pseudonym mechanism is utilized, and an identity-based signature is used to achieve the authentication of vehicles. In [117] author proposed reputation management for a cloud-based IoV system. Routing based metrics are being used to monitor the vehicles and, based on that, update their reputations. Based on the reputation, a certificate is issued to show trust in the vehicular system.

Many researchers have integrated blockchain to address the centralization issue in smart grids. Liang et al. [118] enhanced the smart grid's defence against cyberattacks using a blockchain-based security framework. However, it's not fit for lightweight devices like smart meters due to its high processing time, computational cost, and consensus delay. Jindal et al. [119] introduced a blockchain-based smart grid scheme called "guardian," but its PoW consensus algorithm has a prolonged confirmation time, making it unsuitable for smart grid use. Wang et al. [120] presented anonymous authentication with key management, but its cryptographic approach is computationally intensive for devices like smart meters. While these methods address the smart grid's centralization, their consensus delays are too long, and throughput is too low for optimal smart grid operations. Wang et al. [121] suggested using smart contracts and delegated consensus mechanisms for smart grid trading, emphasizing fairness and a reputation system for P2P trading. However, its consensus algorithm, similar to PoW, is computationally intensive. Guan et al. [122] introduced a privacy-preserving energy trading scheme using blockchain and attribute-based encryption. They also presented a credibility-based equity-proof consensus algorithm to boost efficiency.

Yang et al. [123] introduced a consortium blockchain trading mechanism for P2P energy trading based on a PoS protocol. They employed smart contracts and launched a cryptocurrency named "elecoin" for the P2P market. Similarly, Cao et al. [124] presented a solution addressing credit and security concerns in energy trading, utilizing elecoins to optimize the utility of energy traders. Their approach models the elecoins-purchasing dilemma as a hierarchical Stackelberg game. A miner reward mechanism was suggested in [125] for energy trading. This paper recommends a consortium blockchain system using a PoS consensus algorithm to ensure secure, transparent energy transactions and reduce trading costs. They also introduced the elecoin cryptocurrency for P2P trading within microgrids. These state-of-the-art schemes are prone to various attacks in the blockchain application environment. Karumba et al. [126] introduced HARB, a hypergraph-based framework for energy trading using consortium blockchain. HARB's three-layered network architecture addresses scalability and distributed trust issues in energy trading communities. AlSkaif et al. [127] introduced a peer-to-peer energy trading strategy for residential systems, determining bilateral trading preferences based on surplus power matching and trader distance. Ping et al. [128] presented a privacy-preserved Optimize Energy Trading (OET) scheme using blockchain. They proposed a privacy-

preserving byzantine fault tolerance coordination algorithm to account for dishonesty and ensure accurate trading results. Christidis et al. [129] introduced a framework for creating a blockchain-based local energy market, highlighting that the blockchain type significantly impacts energy distribution and trading performance. However, this framework faces scalability issues. Guo et al. [130] proposed an architecture for distributed energy trading on a byzantine-based blockchain, leveraging the PBFT consensus algorithm and smart contracts. However, the PBFT-based model is deemed inefficient due to high message overhead. Many state-of-the-art schemes used auction mechanisms to further enhance energy trading in the smart grid.

Nguyen T. et al. [131] discusses an Ethereum blockchain-based framework for iterative double auction that's trustless and decentralized. It boasts low latency and fee-less on-chain transactions. Chen Z. et al. [132] introduce a fair cloud auction design using blockchain, achieving financial fairness through a timed commitment scheme with deposits. However, this design exposes other parties' bids, leading to potential bid-rigging. Generally, blockchain and double auction systems risk exposing sensitive buyer and seller information. In contrast, our auction approach in the proposed scheme ETradeChain is budget-balanced, fair to all participants, lightweight, and decentralized in bid allocation, relying on the entire peer network. Singla S. et al. [133] also used the Virkrey-Clarke-Groves Auction for efficient energy trading along with providing the fair optimal pricing algorithm that does not discriminate and increases social welfare. However, it also has a centralized auctioneer, which can have a single point of failure problem. Ma et al. [134] used a blockchain-based distributed Feedback-based Combinatorial Multiunit Double Auction (FC-MDA) mechanism for trading the resources of the cloud. The state-of-the-art schemes faced performance bottlenecks due to design issues in the PBFT and other modified versions of PBFT. The issues are unbiased leader selection, improper reward distribution, and high message overhead complexity. Therefore, to solve the issues, we have proposed various schemes. For IoT applications, we have proposed BLISS, BSS, and SCAB-IOTA, which solve the issues of authentication and secure communication of devices. For ITS, we have proposed LBSV and SmartCoin, which provide secure communication and enable efficient micropayments within intelligent transportation networks. For the Smart Grid, SmartGrid-NG, ETradeChain, and $zk-SGB$ were proposed to facilitate secure energy trading and efficient grid management. Each technique was designed to address specific challenges within its respective domain, demonstrating the versatility and potential of blockchain in improving various aspects of smart city infrastructure.

Concluding Remark

In this chapter, we comprehensively reviewed the existing literature on blockchain scalability, interoperability, and applications, identifying the limitations and gaps in current research. We explored various consensus algorithms, such as PoW, PoS, PBFT, and DPoS, and discussed their strengths and weaknesses. Additionally, we examined the current state of blockchain applications in domains like IoT, ITS, and smart grids. The findings highlighted the need for innovative solutions to address the scalability and interoperability challenges that hinder the widespread adoption of blockchain technology. These insights form the foundation for the subsequent chapters, where we propose novel techniques to overcome these challenges. In the next chapter, we focus on enhancing the scalability of blockchain networks through the development of BlockTree and UnoShard, which aim to improve transaction throughput and overall efficiency.

3

Scalable Blockchain

Bitcoin is the first decentralized system that provides users with virtual currency to buy and sell services and goods. Since then, Bitcoin has autonomously evolved because of its properties of resiliency, integrity, and transparency. However, Bitcoin suffers from scalability issues, which restrict its wider adoption. Scalability in blockchain refers to the capability of a blockchain network to handle a growing amount of transactions or workloads efficiently. It's an essential aspect that determines how well a blockchain can grow and adapt to increased demands without compromising performance and security [135]. Scalability is a key area of research and development within the blockchain community, as improving it is crucial for the widespread adoption and practical use of blockchain technology in various applications beyond cryptocurrencies [23]. Over the past years, the Bitcoin community has witnessed major divisions (forks) over how Bitcoin should scale [136]. Consequently, it is a simple tradeoff between scalability and centralization: as the size of the blockchain grows, few devices will have the capacity to store and process the complete blockchain, which leads the network to become more centralized [137]. Scalability has emerged as a significant hurdle as the technology matures and its adoption grows. To address these challenges, sharding, a concept traditionally used in databases, has been adapted to blockchain technology as a promising solution. Sharding partitions the blockchain network into smaller parts called shards, where each shard can process its own transactions. It increases the network's throughput and scalability by allowing multiple transactions to be processed simultaneously. It also enhances decentralization by distributing the computational load across different nodes. Despite the advancements in sharding made by state-of-the-art schemes like ELASTICO [138], OmniLedger [74], RapidChain [78], and Chainspace [80], they encounter limitations such as high cross-shard transaction latency, low parallelism, centralized reconfiguration, and vulnerability to 1% attack, double spending attack, and majority attack. To address these challenges, we proposed two solutions, namely, *BlockTree* and UnoShard. In this chapter, we first explore the challenges in achieving scalability, sharding in blockchain, and the proposed *BlockTree* and *UnoShard* solution for scalable blockchain.

3.1 CHALLENGES IN ACHIEVING SCALABILITY

There are three major challenges in achieving scalability in blockchain: satisfying blockchain trilemma, synchronization and high latency, and performance concerns. Below, we have defined these challenges in detail.

- **Blockchain Trilemma:** Scalability, security, and decentralization are known as the blockchain trilemma. The significance of this trilemma is that enhancing one aspect leads to compromises in other aspects. For instance, enhancing the scalability by adding more transactions per block can compromise security and decentralization. It is a tradeoff between scalability and centralization: as the size of the blockchain grows, few devices will have the capacity to process the complete blockchain, which leads the network to become more centralized [74].
- **Synchronization and High Latency:** Synchronization involves ensuring that all nodes in the blockchain network have the same, most current version of the ledger, whereas latency refers to the time it takes for a transaction to be confirmed and reflected across all nodes, in the network. Ensuring the ledger synchronization across all the nodes in the network with high latency or consensus delay is a big challenge in achieving scalability. Lowering the consensus delay depends on the effective design of the consensus algorithm. Therefore, designing an effective consensus algorithm to ensure synchronization with low latency is highly important [139].
- **Performance Concerns:** Ability to process a high volume of transactions within a short period is one of the biggest challenges in blockchain scalability. Conventional blockchains like Bitcoin and Ethereum show very limited performance, such as 4-7 transactions per second and 30-40 transactions per second, respectively, which is significantly lower compared with centralized payment systems like Visa and Mastercard. Thus, scaling the blockchain network to meet global transaction demands without compromising security and decentralization is a big challenge. Apart from transaction processing capacity, the high computational and communication overhead also creates a hurdle in scaling the blockchain performance [21].

3.2 BLOCKCHAIN SHARDING

Sharding is a widely adopted method to achieve high scalability in blockchain. In sharding, the blockchain network is split into smaller networks known as shards, where each shard processes its transactions independently. This means transactions can be processed in parallel, significantly increasing throughput [26]. As

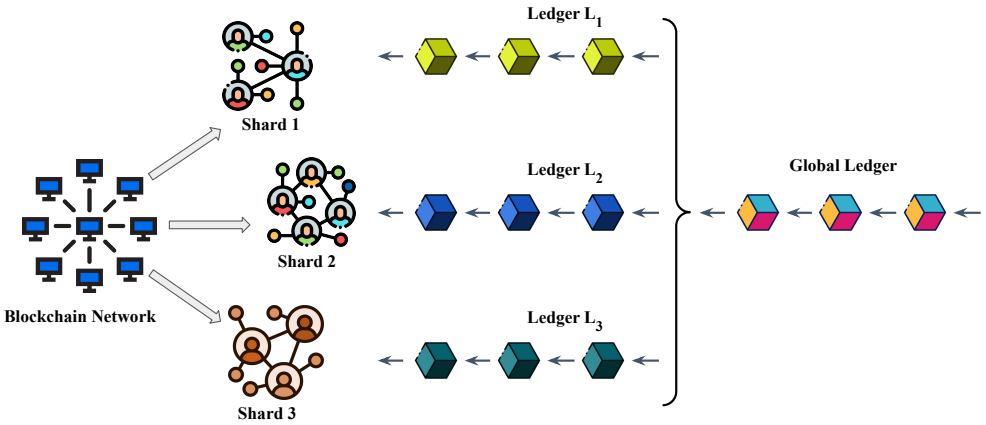


Figure 3.1: An example of a sharded blockchain with three shards. Each shard has its own local ledger L_1, L_2, L_3, and one global ledger.

shown in Figure 3.1, in a sharded blockchain, nodes are allocated into different shards. Each shard maintains its own copy of the ledger (local ledger), i.e., L_1, L_2, L_3 that contains transactions. Apart from these local ledgers, one global ledger is shared with all the shards, which contains all the transactions shared amongst the shards. A transaction may involve a single shard or multiple shards. If a transaction involves a single shard, it is called intra-shard communication, and if it involves multiple shards, it is called a cross-shard transaction. To execute and confirm the cross-shard transaction, nodes in multiple shards communicate with each other.

3.3 CONTRIBUTIONS

While blockchain sharding is an intuitive principle, designing a blockchain sharding protocol is challenging. Sharding blockchain introduces consistency issues across shards. In addition to this, cross-shard communication introduces additional delay, which further reduces the system throughput. In the first chapter, we have listed the key issues of blockchain sharding that need to be solved to enhance the system's performance. To solve these issues, BlockTree opted for two approaches: first, minimizing the constraint that each transaction is broadcasted and processed by the entire network, and second, restructuring the relationship between blocks from a chain to a graph. On the other hand, the UnoShard works in three different phases, i.e., fair shard allocation, intra-shard consensus, and cross-shard consensus. The shard allocation phase leverages the Delegated Proof of Stake (DPoS) consensus scheme, where the number of shards is equivalent to the number of delegates that have been selected. The intra-shard consensus phase used the concept of a trust-based Practical Byzantine Fault Tolerance (PBFT) scheme, where the trust is calculated based on the direct interaction between the nodes and the feedback from the nodes. The block generated by each shard is named Local Block (*LB*). The last phase is the most critical in every sharding scheme: cross-shard consensus. The cross-shard consensus gathers all the local blocks and generates a Global Block (*GB*) for the global ledger, which is shared amongst all the peers in the network. Following are the major contributions of BlockTree and Unoshard.

- **Restructured Blockchain:** Proposed scheme, BlockTree, a restructured blockchain design from linear to nonlinear structure that can process multiple transactions in parallel and serves the purpose of sharding transactions.
- **Consensus Algorithm:** Apart from restructuring blockchain, we proposed a novel leader selection and consensus algorithm. To mark the new leader selection in the shard, we have created two different types of blocks, i.e., data block and leader block. The leader block is created when a new leader is selected, and that leader creates the data blocks, which contain the new set of transactions.
- **Cross-Shard consensus:** In UnoShard, we have proposed a novel cross-shard consensus mechanism, which creates the global ledger in cooperation with all the shards to ensure zero possibility of double-spending and fault tolerance. To establish cooperation among all the shards, cross-shard consensus used a novel concept called vote chain, which ensured the confirmation of all the transactions processed by each shard and bunched in the form of a local block.
- **Trusted Intra-Shard Consensus:** The proposed UnoShard leverages the trust-based PBFT consensus mechanism in each shard to confirm the transactions and generation of local blocks. The trust value of each node is computed based on the comprehensive evaluation of the node's historical per-

formance and behavior within the network. This includes factors like the node's accuracy in transaction processing, consistency in participating in consensus activities, uptime, and overall cooperative behavior. The node which has the highest trust value is called the leader node.

- **Shard Allocation:** In addition to cross-shard and intra-shard consensus methods, we designed the shard allocation method based on the delegated proof of stake (DPoS) method. The number of shards is equal to the number of delegated in the selected in the first phase of the UnoShard scheme.
- **Performance and Security Analysis:** We have conducted a comparative comprehensive performance analysis of BlockTree based on the parameters, throughput, consensus delay, transaction frequency, fairness, and leader selection time. We have also performed a security analysis of the BlockTree and UnoShard based on fork prevention, scalability, double-spending, 1% attack or shard takeover attack, and fault tolerance prevention. Both analysis shows the superiority of BlockTree and UnoShard over the state-of-the-art schemes.

3.4 BLOCKTREE: A NONLINEAR STRUCTURED, SCALABLE DISTRIBUTED LEDGER

The BlockTree is a blockchain protocol that offers better and improved consensus delay and throughput than protocols such as Bitcoin-NG and Algorand without losing the security. This protocol improved in two ways: the first structure of the blockchain and the second consensus algorithm. The structure of a BlockTree is no longer a chain structure but follows a tree structure (m-way tree structure). The protocol splits the network based on the number of miners in the network. Here, "m" defines the upper limit of the number of miners in the network. If, at a particular time, the number of miners increases more than m (i.e., $number_of_node > m$), it splits the network from that instant, i.e., a new shard is created. This process is repeated in every group, and it grows like an m-way tree. As soon as the network is split and a new shard is created, the current blockchain also gets forked, and the newly created shard starts mining and adding the block into this new branch of the chain. This way, we can employ the parallel execution of transactions, increasing the throughput and decreasing the consensus delay. Figure 3.2 is the structure of the BlockTree. Every miner node keeps track of a number of miners in the group with parameters $number_of_node$, and *Algorithm* 1 keeps working continuously and checks $number_of_node > m$ for creating a shard. This parameter is updated every time a new miner is added to the network. Furthermore, in each newly created shard, a group leader is selected periodically, i.e., there is a fixed time interval between selecting any two group leaders. *Algorithm* 2 described the process of leader selection. The leader is responsible for the serialization of transactions generated in that interval. The protocol generates two types of blocks: leader block and data block. The leader block is added when a new leader is selected, and after this leader block, the data block is added based on the consensus algorithm. The leader block contains no transactions, while the data block contains the transactions. The leader block acts as a mark of a leader, i.e., it defines which miners add the data blocks. If it is found that some transactions are being forged or modified, then the leader node is marked as a bad miner so that it cannot take part in the next group leader selection. Thus, the probability of the current leader getting selected is minimal so that all the miners of the group can get an equal opportunity to become a leader.

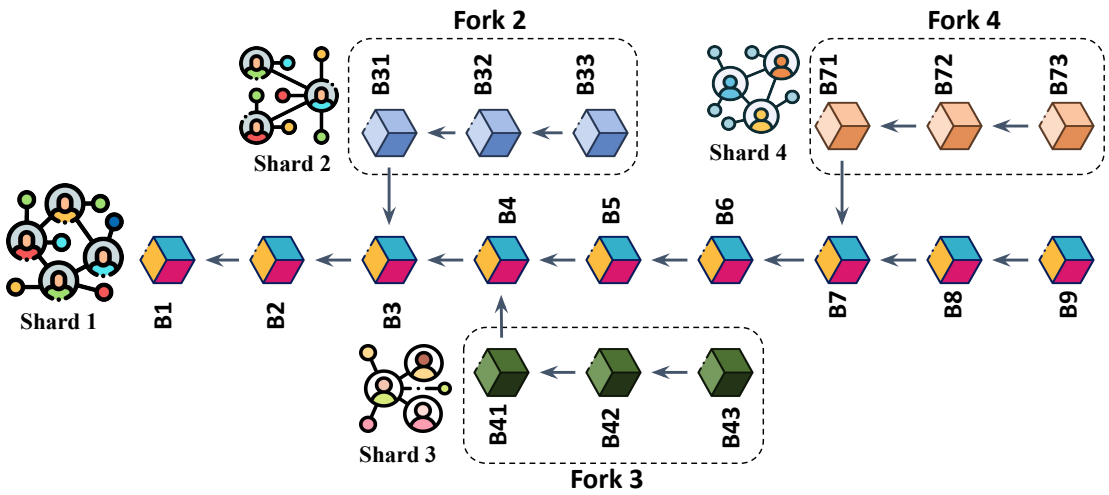


Figure 3.2: Struture of BlockTree

3.4.1 Sharding in BlockTree

As we have mentioned in the previous section, when the condition $number_of_node > m$ (i.e., the count of miners is greater than the set limit m) is true, a shard is created in the blockchain. This newly created shard included the recently joined miners. Therefore, any miner who joins the network must keep the time of joining the network. Every leader block also includes the time of joining the network for miners. In each shard, for generating and verifying the data blocks, BlockTree uses the *Algorithm* 3. After verification, the leader adds the data blocks to the chain. Since we have employed the consensus algorithm for leader selection and the data block verification, the BlockTree does not suffer from illegitimate fork generation. Due to the inclusion of two consensus algorithms, BlockTree also does not suffer from the double-spending problem. *Algorithm* 1 describes how a shard is created in the BlockTree. This algorithm continuously works at all the shards of the blockchain network and checks how many miners have been added. *Algorithm* 1 works as follow:

1. This is a continuous process until the system functions properly. Since it is a public blockchain, any node can join and leave the network. Any node who wants to be a miner is added into the current group of miners as current_miner_group.add (new_node) after checking if $number_of_node < m$ is true or not. If it is true, then *new_node* is added, and all the nodes follow the *Algorithm* 2 for the leader selection and *Algorithm* 3 for the data block creation and verification.

2. But if the condition $number_of_node < m$ is false, the current shard has more miners than the limit m so that the shard is split with minimum size (size varies from system to system). This newly created shard contains the recently joined nodes.

3.4.2 Transaction Sharding

The BlockTree is a sharding-based distributed ledger protocol. The structure of the scheme is modified so that, instead of all the miners mining one block, we have divided the miners into shards so that many groups of miners (shards) can mine the blocks parallelly. This change in structure and sharding of transactions

Algorithm 1: Algorithm for Shard Creation

Input: New_Node (Node's Public Key and Joining Time)
Output: Shard Creation
Function `createShards(`*nodes*`):`

```
while true do
    if number_of_node < m then
        current_miner_group.add(new_node);
        // Follow the Algorithm 2 to select the leader and Algorithm 3 to
           create and verify the data blocks
    else
        create_shard(minimum_Group_Size);
        current_miner_group.shard(all_recent_new_node)
```

improved the throughput and the consensus delay. One more parameter that is enhanced due to this sharding is the scaling of the blockchain network, i.e., BlockTree is the highly scalable protocol compared to the state-of-the-art schemes. According to the scheme, anyone can join the network, and this does not cause any overhead on the network because as soon as the number of miners increases from the defined limit, the network is split, and a new shard is created that mines the block on the forked chain. Hence, due to the structural change, the performance is improved in terms of throughput and consensus delay, and the scalability issue is also resolved.

3.4.3 Leader Selection, Leader Block, and Data Block

The leader blocks are used to mark the leader selection. The leader block is similar to the Bitcoin block, except that it does not contain any transaction value. Every leader block must contain miner ID, block number, timestamp, coinbase transaction, and nonce value. A leader block is valid if its hash is smaller than the threshold value. It also contains the leader's public key because this public key is used to verify all the subsequent data blocks until the new leader block is added. The time is equally divided among all the leader blocks to maintain a uniform rate of data block generation. Leader selection in BlockTree is different from Bitcoin-NG in that the probability of selecting the current node as the leader node is minimal, i.e., unlike Bitcoin-NG, BlockTree is fair, where all nodes have an equal chance to become a leader. The two parameters of the algorithm, status and age, are essential for leader selection to be fairer. The status tells about the behavior of the node because if any node is found malicious, its status is set to bad, and the age parameter tells whether it is a current leader or not. *Algorithm* 2 for leader selection works as follows:

1. This process is followed in every group of the BlockTree blockchain network. All the nodes participate in the leader selection process. For this, every node should have the timestamp T, previous hash block (PBH), which is calculated by the function $getHash()$. Though every node has a copy of the current blockchain, using the $getHash()$ function gets the last block's hash. Along with the timestamp and hash of the previous block, it set a *nonce* value to 0.

2. A threshold hash value $thresholdValue$ is set, and every node starts calculating the target hash value $targetHash$, which should be less than the $thresholdValue$. Node uses the function $calculateHash()$ to calculate the hash value. Calculating the $targetHash$ value also checks the status and age of the

node for fair selection and eliminates malicious nodes. The node with the least age has the highest preference for leader selection.

3. After the leader node selection, a leader block (*leaderBlock*) is created using block creation function *createBlock*() with parameters, T, *minerID*, *PBH*, *status*, *age*, *block_number*, *targetHash* and appended to the existing ledger.

Algorithm 2: Leader Selection Algorithm

Input: Node[i], i=1,2,3,.....
Output: Leader Node
Function `leaderSelection(`*nodes*`):`

```
while LeaderIsSelected == 0 do
    for each node[i] in group do
        T = generate_unique_timestamp()
        nonce = 0
        targetHash = calculateHash(T, nonce, previousBlockHash, node[i].ID)
        while targetHash >= thresholdValue do
            nonce+ = 1
            targetHash = calculateHash(T, nonce, previousBlockHash, node[i].ID)
        if status! = bad&age(node[i]) < age of all the nodes then
            node[i] = leader
            LeaderIsSelected = 1
            broadcastLeaderSelection(node[i].ID)
if node[i] == leader then
    leaderBlock = createBlock(T, PBH, minerID, status, ageblock_number, targetHash);
    blockchain.append(leaderBlock);
```

There are two types of nodes in the blockchain network: one that generates the transactions, usually called ordinary nodes, and the second type of node, called miner node, which does the mining work. One of the miner nodes is elected as the leader node following the leader selection algorithm, and the leader block is generated to mark the entry into the blockchain ledger. A leader is allowed to generate data blocks until the next leader is selected. The time between the two leader blocks is the interval for data block generation. The size of the data block is predefined. The data block content is ledger entries, Mrt, T, and PBH. The leader's private key signs the data blocks, which are verified by the other miners (peer nodes). Due to timestamp and status parameters, the malicious data block is prevented from being added to the chain. Figure 3.3 shows the arrangement of a data block and a leader block. The leader node collects all the recently added transactions from the transaction pool. Compute the Mrt and then compute the data block, including T, PBH, and *nonce* value. After creating the data block, the leader signs it and sends it to other nodes in the shard along with the *nonce* to verify it. Each non-leader node independently verifies the block and then broadcasts a "valid block" message to all nodes, including both the leader and the other non-leader nodes. To verify the data block, each non-leader node calculates the Merkle root and checks whether the received block and calculated block are the same or not. If both are the same, then set the block status as valid; otherwise, set it as invalid and set the status of the leader node as bad so that it will not be selected again as a leader node. By broadcasting the "valid block" messages to all nodes, the protocol guarantees that the consensus process is transparent and that all nodes can participate in verifying and accepting the

leader block. This approach prevents any single point of failure and ensures that the blockchain remains secure and decentralized. If the received valid block status is more than $2K+1$, the block is added to the chain; otherwise, it is discarded (where K is the number of faulty nodes). *Algorithm* 3 shows the generation and verification of data blocks. A detailed description of the *Algorithm* 3 is as follows:

1. All the ordinary nodes connected to the miner nodes generate the transactions and put them in the transaction pool (TP). All the miner nodes share the transaction pool. The leader node collects all the transactions from the TP and then, after calling the function $createHashTree$, calculates the Mrt and passes these transactions.

2. After calculating the Mrt, it also calculates PBH using the function $getHash$ and recod the current timestamp T.

3. After calculating Mrt, PBH and getting timestamp T and $nonce$, it creates a data block somewhat different from the leader block. The data block is created using the function $createBlock()$ with the parameters described above. After creating the block, the leader node signs it using its private key and broadcasts it into the network for verification.

4. The non-leader nodes receive the data block and verify using the public key of the leader. These nodes also calculate the Mrt from the TP, create the data block, and check whether the received block matches with the regenerated block. If both match, then set the $block_status$ as $valid$; otherwise, set it to $invalid$ and set the node status $node.status$ as bad so that in the next leader selection, it will not be selected and considered as the malicious node.

5. If the leader node receives at least $2K+1$ $valid$ response, it adds a data block into the blockchain ledger.

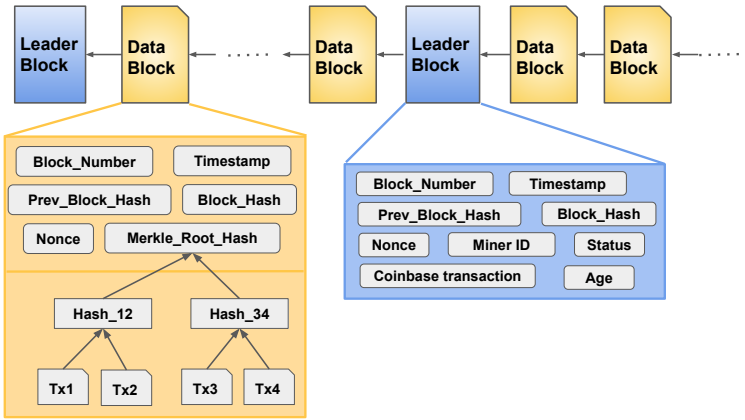


Figure 3.3: Types of blocks and their arrangement

Total Ordering of Transactions

In BlockTree, the total ordering of transactions is achieved through a combination of linear ordering within each shard and timestamp-based ordering across the entire network. Below is a detailed description of how the ordering of transactions is achieved in BlockTree.

Algorithm 3: Data Block Verification Algorithm

Input: Transactions: $Tr_1, Tr_2, ..., Tr_n$
Output: Data_Block
Function `dataBlockVerification(`*nodes*`):`

```
TP.append = new_transaction;
if node == leader then
    Mrt = createHashTree(TP.getAllTransactions);
    T = Timestamp;
    nonce = random_number;
    PBH = getHash(previous_block);
    BH = getHash(Mrt, T, PBH);
    block = createBlock(Mrt, T, PBH, BH, nonce, glocalTimestamp, node[i].ID);
    sign_block(block);
if node != leader then
    receive_block(block, nonce);
    Mrt = createHashTree(TP.getAllTransactions);
    block* = createBlock(Mrt, T, PBH, BH, nonce, glocalTimestamp, node[i].ID);
    if block == block* then
        block_status(valid);
    else
        block_status(not_valid);
        node.status = bad;
if (count(block_status == valid) ≥ 2K + 1) then
    blockchain.append(block);
    synchronize_timestamps(block);
```

1. **Linear Ordering within Shards:** Each shard in the BlockTree structure functions like a traditional blockchain, maintaining a linear sequence of blocks. Within a shard, the leader node generates a leader block to mark the beginning of its term, followed by a sequence of data blocks that contain transactions. These blocks are linked sequentially, referencing the previous block's hash, thereby creating a linear chain of blocks. This method ensures that transactions within a shard are processed in a clear, sequential order, similar to the traditional blockchain approach used in Bitcoin. Transactions are included in the blocks in the order they are received and verified by the leader. This ordering ensures that all transactions within a shard are added to the ledger in a consistent and linear manner. The process of adding transactions to blocks, verifying them, and appending the blocks to the chain maintains this linear order. As each shard operates independently, the integrity and order of transactions within a shard are preserved, providing a clear and unambiguous sequence of events for all transactions processed within that shard.

2. **Timestamp-Based Ordering Across Shards:** To maintain synchronization and achieve a total order of transactions across different shards, BlockTree employs a timestamp-based ordering mechanism. Each transaction is assigned a global timestamp when it is created, reflecting the time at which the transaction was first generated and broadcast to the network. These timestamps are crucial for ordering transactions across shards, ensuring that even though transactions are processed in parallel

within different shards, they can be consistently ordered across the entire network. Periodic synchronization points are established across shards, during which leaders communicate and reconcile the order of transactions based on their timestamps. When transactions are verified and added to the blocks, their global timestamps are used to order them relative to transactions in other shards. This timestamp-based approach ensures that the entire network maintains a coherent and synchronized order of transactions, despite the parallel processing capabilities provided by the m-way tree structure. By leveraging timestamps, BlockTree ensures that all transactions are integrated into a single, unified ledger, providing a comprehensive view of the transaction history across all shards.

3.4.4 Remuneration Model

In the case of Bitcoin, a miner is compensated for the efforts to mine the block in the form of some bitcoins. Similarly, BlockTree also has remuneration for mining the block to motivate the mining. A leader is compensated for adding the data blocks into the chain. The remuneration is compensated based on the data blocks added by leaders, i.e., the addition of each data block carries a fee, which would be called a transaction fees. BlockTree has a transaction fee as a remuneration. If the number of data blocks added by the leader node is 10, each data block's transaction fee depends on how many transactions it contains. For example, if every data block contains 200 transactions on average, ten blocks would contain 2000 transactions. Hence, the general equation for transaction fees equals to

$$Total_fee = TF \sum_{i=1}^{n} DB_i \tag{3.1}$$

where DB_i is the number of transactions in i^{th} data block and TF is a transaction fee of each transactions.

3.5 UNOSHARD: HARMONIZING SCALABILITY AND SECURITY

BlockTree introduced a novel approach to blockchain scalability by restructuring the blockchain into an m-way tree, enhancing parallel transaction processing. However, despite its innovations, BlockTree faces challenges related to performance, double-spending, and certain security vulnerabilities. To address these issues, we developed UnoShard, a more advanced sharding technique. Unlike BlockTree, UnoShard employs a trust-based PBFT consensus mechanism within each shard and introduces a novel cross-shard consensus mechanism called VoteChain. These enhancements significantly improve performance, prevent double-spending, and strengthen security against various attacks. Therefore, UnoShard offers a more robust solution to the challenges of BlockTree and other state-of-the-art schemes for blockchain scalability and security.

UnoShard is a novel approach that harmonizes scalability and security using the sharding technique. The UnoShard works in three different phases, i.e., fair shard allocation, intra-shard consensus, and cross-shard consensus. The shard allocation phase leverages the DPoS consensus scheme, where the number of shards is equivalent to the number of delegates that have been selected. The intra-shard consensus phase used a trust-based PBFT scheme, where the trust is calculated based on the direct interaction between the nodes and the feedback from the nodes. The block generated by each shard is named Local Block (*LB*). The last phase is the most critical in every sharding scheme: cross-shard consensus. The cross-shard consensus gathers all the local blocks and generates a Global Block (*GB*) for the global ledger, which is shared amongst all the

peers in the network. To combine all local blocks and verify them, UnoShard uses the concept of Vote Chain (*VC*) and uses the Directed Acyclic Graph (DAG) to construct the root hash of all the transactions instead of a merkle root tree for *GB*. To accomplish these tasks, all the shards select a Shard Coordinator (*SC*), usually called an inter-shard coordinator. The *SC* is selected based on the major three critical elements, i.e., accumulative reputation score (ARS), decaying reputation score (DRS), and performance metrics-based reputation score (PMRS). These comprehensive approaches ensure that UnoShard effectively addresses the complexities of sharding in blockchain networks. Further, it is important to note here that UnoShard leverages the UTXO (Unspent Transaction Output) model as the foundation for its transaction processing. The UTXO model, widely used in blockchain networks like Bitcoin, maintains the state of the blockchain through unspent transaction outputs. This model enables efficient transaction validation and supports the scalability and security enhancements that UnoShard aims to achieve.

UnoShard addresses the issues of 1% attack, double spending attack, and majority attack, thus enhancing the overall security and robustness of the blockchain network. By overcoming the limitations of existing sharding schemes, UnoShard presents itself as a robust, scalable, and secure framework, paving the way for the next generation of blockchain applications. Details of these phases are given in subsequent sections.

3.5.1 Shard Allocation

The *Algorithm* 4 outlines the process of distributing nodes between shards in a blockchain network. It begins by initializing arrays for vote weights and delegate votes based on the total number of stakeholders and delegates. The functions of the *algorithm* include *calculatingVoteWeights*, *electingDelegates*, and *nodeAllocation*. There are five key steps based on which the shard allocation is performed in UnoShard. Each of these key steps plays a critical and significant role in ensuring fairness, efficiency, and decentralization in the network. The key steps are as follows.

Computing Weighted Vote

The vote of each stakeholder is weighted based on the stakes they put in to be a part of the active working node in the network. The following equation (4) is followed to compute the weight of every stakeholder's vote.

$$W_i = \frac{s_i}{\sum_{k=1}^{M} s_k} \tag{3.2}$$

where W_i is the weight of the stakeholder's i vote, S_i is the stake of the corresponding stakeholder, and M is the total stakeholders. It is important to note that the stakeholder is nothing more than the consensus nodes in the blockchain network. It is obvious that the stakeholders with the highest stakes in the network have a higher proportional impact on the delegate selection process. The function *calculatingVoteWeights* in *Algorithm* 4 performs this computation.

Delegate Selection and Shard Formation

The second step in the allocation of the shards is the selection of the delegates after computing the weight of the votes of the stakeholders. Let us say V_d is the sum of the total weighted votes for each delegate. Then, $D \in D_{Max}$ is the number of top nodes with the highest V_d selected as delegates. This process ensures an unbiased and fair selection of delegates, which is very important for any shard-based blockchain system.

The third step is the formation of shards. In this step, each delegate $D \in D_{Max}$ is assigned to a shard. This assigned delegate node is also selected as the block proposer or shard leader for the first round of consensus in the intra-shard consensus process. This second and third step is illustrated in *Algorithm* 4 in function *electingDelegates*.

Nodes Allocation to Shards

After assigning one delegate to each shard, the fourth step is to assign a number of nodes to each shard. We have used the following equation (3.3) to allocate the number of nodes to each shard evenly.

$$NNS = \frac{N}{D} \tag{3.3}$$

NNS denotes the number of nodes in each shard, N is the total number of nodes in the network, and D is the total number of delegates selected in the first step. Thus, each shard contains *NNS* nodes to maintain the true balance and efficiency of the network operation. The function *nodeAllocation* is dedicated to this fourth step of shard allocation.

Periodic Delegate Re-election and Re-assignment

New delegates are re-elected and re-assigned after every 100^{th} *GB* in the *GL*. It is an important step to prevent the system from becoming biased and unfair in transaction processing. It also plays an important role in preventing the shard takeover attack. Thus, this step is crucial for the efficient and fair functioning of the blockchain system. To do this, all the functions of *Algorithm* 4 are called again in the correct sequence, i.e., first, *calculatingVoteWeights*, second *electingDelegates*, and third *nodeAllocation*.

3.5.2 Intra-Shard Consensus

This is the second phase of UnoShard. After the successful completion of the shard allocation phase, the intra-shard consensus takes place. The intra-shard consensus is essentially a trust-based PBFT consensus *algorithm*, where the block proposer is selected based on their trust value. As a novel contribution, we have derived an efficient trust computation scheme for the nodes in the shard. This trust computation is important for attack-free transaction processing and verification. Trust assessment among the nodes in the shard is crucial for maintaining a consistent and stable view of the shard. The intra-shard consensus process is again divided into five steps, namely, initial trust assignment, trust score computation, trust score normalization, and PBFT consensus process.

As stated in the Shard Allocation section, during the 'delegate selection and shard formation' step, the top nodes with the highest weight are selected as delegates and are assigned to each shard. This assigned delegate node is also selected as the shard leader for the first round of the consensus. Therefore, in this step, the initial trust score $N(T)_i$ of all the nodes is set to *zero* except the assigned delegate node. That is, let's say N_k is the delegate node, then the trust score $N(T)_k$ of this node is set to 1, and the trust score of other nodes is set to 0. This can be represented as follows:

$$N(T)_i = \begin{cases} 1 & \textit{if } i = k \textit{ is the delegate node} \\ 0 & \textit{otherwise} \end{cases} \tag{3.4}$$

Algorithm 4: Algorithm for Shard Allocation

Input: N (Total number of nodes), D (Total number of delegates), s (Array of stakes, size M), M (Total number of stakeholders)
Output: shardAssignments (Array of nodes assigned to each shard)
Initialize array W of size M for vote weights;
Initialize array V of size D for delegate votes;
Function `calculatingVoteWeights()`:
    $totalStake \leftarrow \text{sum}(s[i] \text{ for } i = 1 \text{ to } M)$;
    for $i = 1$ *to* M **do**
        $W_i = \frac{s_i}{\sum_{k=1}^{M} s_k}$;
Function `electingDelegates()`:
    for *each delegate d* **do**
        $V[d] \leftarrow \text{sum}(W[i] \times$ *votes for d from stakeholder i*$)$;
    selectedDelegates $\leftarrow$ *top D indices in V with highest values*;
Function `nodeAllocation()`:
    nodesPerShard $\leftarrow N/D$;
    currentNode $\leftarrow 1$;
    for *each shard in selectedDelegates* **do**
        shardAssignments$[$*shard*$]$ $\leftarrow$ *list of nodes from currentNode to currentNode + nodesPerShard* - 1;
        currentNode $\leftarrow$ *currentNode* + *nodesPerShard*;
Call `calculatingVoteWeights()`;
Call `electingDelegates()`;
Call `nodeAllocation()`;

After the first round of the consensus process, the trust score of all the nodes is updated. The trust score update depends on two important parameters: 1) the node's historical performance and 2) behavior within the network. These parameters further include factors like accuracy in transaction processing from node and network inclusion time, i.e., up-time in the network. The trust score update process ensures that nodes within the shard maintain a consistent and reliable view of each other's trustworthiness. While it is not required for all honest nodes to have identical trust metrics, the continuous feedback and interaction mechanism lead to a convergence of trust scores over time. This dynamic and decentralized approach ensures that the trust metrics used in the consensus process accurately reflect the overall behavior and reliability of the nodes, thus maintaining the security and efficiency of the blockchain network. Mathematically, this is represented as:

$$N(T)_i^{(new)} = \sum_j N(BV)_{ji} * N(T)_j^{(old)} \tag{3.5}$$

Here, $N(BV)_{ji}$ denotes the behavior value given by node N_j to node N_i, reflecting node N_j's assessment of node N_i's reliability. The $N(T)_j^{(old)}$ is the previous trust score of node N_j. This calculation aggregates trust feedback from multiple nodes to revise $N(T)_i^{(new)}$. The current shard leader prepares the 2X2 reputation

matrix for the C_{ji} as follows:

$$\begin{bmatrix} N(BV)_{1,j}^{(k)} & , & N(BV)_{1}^{(k-1)} \\ N(BV)_{2,j}^{(k)} & , & N(BV)_{2}^{(k-1)} \\ & \vdots & \\ N(BV)_{i,j}^{(k)} & , & N(BV)_{i}^{(k-1)} \end{bmatrix}$$

Here, $N(BV)_{1,j}^{(k)}$ depicts the behavior value of N_j on N_1 in the k^{th} round of consensus, and $N(BV)_{i}^{(k-1)}$ is the behavior value of N_i in the $(k-1)^{th}$ consensus round. To achieve effectiveness and accuracy in trust score computation, the trust score values computed above in equation (3.5) are normalized. Normalization involves adjusting the computed trust score on the 0-1 scale, where 0 means the lowest trust score and 1 means the highest trust score. This normalization step is essential for removing any possibility of biases from historical performance and behavior analysis and preserving the scoring consistency.

The node with the highest trust score is selected as the shard leader for the next consensus round. In the last step of the intra-shard consensus, the shard leader collects transactions from the global transaction pool, with each local block capable of including a maximum of 500-800 transactions, and starts the process of generating *LB*. This constraint ensures that each shard's transaction processing load remains manageable and efficient. To generate the *LB*, the intra-shard follows the typical PBFT process after selecting the shard leader. The process of consensus based on PBFT is shown in Figure 3.4. The role of the trust-based shard leader as the block proposer is pivotal in ensuring the security and integrity of the *LB* generation process. Leveraging the high trust for shard leader selection maintains a reliable and efficient consensus among the nodes in the shard. This ensures that the transactions in the shard are processed by the most trusted nodes, which further enhances the robustness of the shard network. The output of this phase is represented as LB_{S_i}, where S_i denotes the i^{th} shard number.

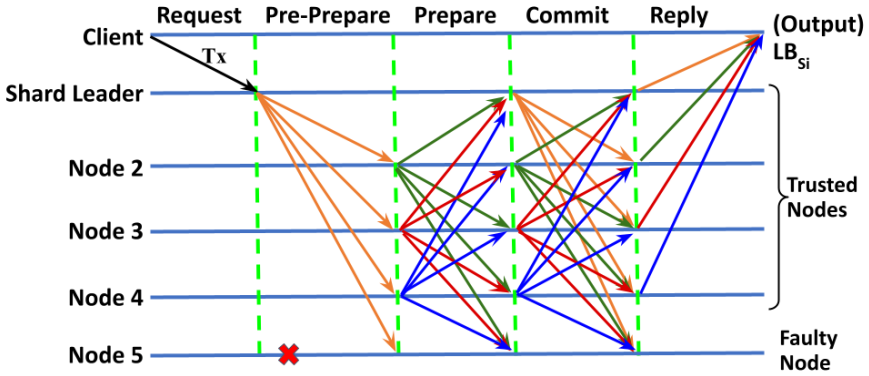


Figure 3.4: Intra-shard consensus process of UnoShard based on PBFT

3.5.3 Cross-Shard Consensus

Cross-shard consensus is the most challenging thing in any sharding scheme of the blockchain network. Therefore, it is also the last and most crucial phase of the UnoShard. This step basically, collects all the local blocks, say $LB_{S_i}, i = 1, 2, ..., k$, where k is the number of shards in the blockchain network and generates

one global block (*GB*) that contains all the transactions from all the local blocks. It is important to note that during the *GB* generation, instead of following the Merkle root tree to get the root hash of the transactions, it uses the DAG structure to be included in the block header. The important purpose of following the DAG over the Merkle root tree is described in the subsequent section. Another key aspect of the Cross-shard consensus is that to establish the cooperation between all the shards for the confirmation of the LB_{S_i}, we have used the concept of VoteChain (VC). That is, VC enables us to synchronize and verify the transactions that are included in the *LB*s from each shard. Moreover, before proceeding with VC and *GB*, this phase selects the inter-shard coordinator from each shard. Thus, the process of this phase is divided into three steps, namely 1) Inter-Shard Coordinator Selection, 2) Creating VoteChain (VC) to confirm all the LB_{S_i}, and 3) creating the *GB* and appending it to the *GL*.

Inter-Shard Coordinator Selection

It is the first step of cross-shard consensus. This step is crucial for the effective and attack-free working of UnoShard's whole sharding process. The inter-shard coordinator selection process is executed based on three important parameters, which are the Accumulative Reputation Score (ARS), Decaying Reputation Score (DRS), and Performance Metrics-Based Reputation Score (PMRS). The whole process of inter-shard coordinator selection is described in the *Algorithm* 5. Because this *algorithm* uses multiple parameters in the computation, we called this a *Multi-Facet Reputation-based Algorithm*. This *algorithm* computes the reputation of each node in the network through a series of reputation factors. The definition of four important parameters is as follows.

A. Accumulative Reputation Score (ARS): This reputation score is calculated based on the node's participation and reliability. This score increases and decreases with the node's positive and negative contribution, respectively. The positive contribution includes successful validation and efficient transaction processing. The value of *ARS* can be 0 or 1, that is, $ARS \in [0,1]$ where 0 means negative participation and 1 means positive participation.

B. Decaying Reputation Score (DRS): This score value represents the participation of the node as active or inactive. The active and inactive participation of the node can be measured based on the response time, which is included in the communication parameter as the *Timestamp*. Again, this can be either 0 or 1, that is, $DRS \in [0,1]$, where 0 means inactive participation and 1 means active participation.

C. Performance Metrics-Based Reputation Score (PMRS): As the name itself described, this score is derived from the addition of quantitative values such as the computation power score (CPS) and the block success rate (BSR). Both the score can be computed as follows:

$$CPS_{N_i} = \frac{Hashing\ Power\ of\ N_i}{Total\ hashing\ power\ in\ the\ network} \tag{3.6}$$

The PMRS incorporates the node's hashing power and block success rate to evaluate its computational efficiency. It is important to note that mentioning "hashing power" in this context does not imply the use of PoW within the UnoShard protocol. Instead, it serves as a performance metric to ensure that nodes involved

in the consensus process have adequate computational resources.

$$BSR_{N_i} = \frac{Number\ of\ Valid\ Blocks\ from\ N_i}{Total\ number\ of\ blocks\ in\ the\ chain} \tag{3.7}$$

After this, the Weighted Sum Calculation (WSC) is carried out as,

$$WSC_{N_i} = CPS_{N_i} + BSR_{N_i} \tag{3.8}$$

Calculate the reputation score ($PMRS_i$) of node N_i using a weighted geometric mean approach (it balances the influence of both parameters) for each node N_i using the finalized reputation calculation method:

$$PMRS_{N_i} = \sqrt{WSC_{N_i}} \tag{3.9}$$

Normalize this value in the range of 0 and 1. That is, this can also be either 0 or 1, i.e., $PMRS \in [0,1]$, where 0 means low performance and 1 means optimum performance.

VoteChain: Confirming All the Local Blocks

VoteChain ensures trust on all the local blocks (LB_{S_i},$i = 1,2,...,k$), where k is the number of shards in the blockchain network. It is an important step towards the finality of the global ledger. A VoteChain comprises linked VoteBlocks, which are generated for all the corresponding *LB*s. Each shard coordinator votes on the validity of each local block generated by each shard. For instance, if there are 20 shards, which generate 20 local blocks, then VoteChain should contain 19 votes on each local block. In this way, we can ensure that every transaction included in *GB* is verified and assigned.

A VoteBlock contains at least $k-1$ voting transactions in each LB_i. Each vote belongs to a unique shard among all k shards. This can be understood as follows. Each VoteBlock corresponds to each local block, i.e., Let us say there are 20 local blocks, then there will be 20 VoteBlock. This illustration is depicted in Fig. 3.5. Further, the VoteBlock constructs a Merkle tree whose leaf nodes are the hash value of each vote transaction generated by the shard coordinators with respect to the corresponding local block. The root of the Merkle tree is included in the header of the VoteBlock. Finally, each VoteBlock is broadcasted to the whole network to help each shard acknowledge the consensus results. Since each VoteBlock is very small in size, it barely affects the storage overhead of the system. Usually, each shard maintains the VoteChain to verify the latest VoteBlocks and make historical consensus results auditable.

All shard coordinators verify the proof of all local blocks generated from all other shards and generate vote transactions as valid local blocks or invalid local blocks. After the intra-shard communication phase, as described above, i.e., transaction verification and recording as a local block, all the corresponding shard coordinators broadcast that local block (LB_i) to other peer shard coordinators for verification and generation of vote transaction to be included in the VoteChain. A local block is to be considered valid if all the shard coordinators confirm the local block as a valid block. When all the local blocks are confirmed, the leader node from the shard coordinator starts the processing of the next phase, i.e., global block generation and final consensus for the global ledger.

It is worth noting that the concept of using vote-based consensus has also been explored in the PRISM protocol [87], which decouples the consensus process into three distinct steps involving proposer blocks, transaction blocks, and vote blocks. The PRISM protocol achieves high throughput and scalability by

Algorithm 5: Multi-Facet Reputation-based Algorithm

Input: AT_i, DT_i, PMT_i
Output: Shard Coordinator Selection based on Reputation Score
Initialization:
for *each node i* **do**
    AT[i] ← 0; DT[i] ← 0; PMT[i] ← 0;
Function InitializeTrustScore(node i);
begin
    AT_i ← InitialATValue, DT_i ← InitialDTValue, PMT_i ← InitialPMTValue;
Function *computeTrustScore*(node *i*, α, β, γ, δ);
begin
    $totalScore \leftarrow \alpha * AT_i + \beta * DT_i + \delta * PMT_i$;
Procedure *SelectLeader*;
begin
    for *each node i* **do**
        score ← *computeTrustScore*(*i*, α, β, γ, δ);
        if *score* > *maxScore* **then**
            maxScore ← score;
            leader ← i;
    return leader;
Procedure *updateTrustScore*(node *i*, AT_Update, DT_Update, PMT_Update);
begin
    $AT_i \leftarrow AT_i + AT_Update$;
    $DT_i \leftarrow \max(0, DT_i - DT_Update)$;// `Decay factor`
    $PMT_i \leftarrow PMT_i + PMT_Update$;
begin
    for *each node i* **do**
        updateTrustScore(*i*, AT_Update_i, DT_Update_i, PMT_Update_i);

employing this multi-block structure. In contrast, UnoShard's approach focuses on sharding the network to process transactions in parallel across multiple shards. While both protocols utilize voting for consensus, UnoShard applies this within the context of sharded local blocks, followed by the aggregation of these blocks into a global block. This distinction shows UnoShard's unique method of leveraging sharding and VoteChain to enhance blockchain scalability and security.

Final Consensus: Global Block Creation and Ledger Updation

In a sharded blockchain network, each shard processes transactions independently, creating local blocks. A leader node then aggregates these local blocks to form a global block. However, this structure introduces the possibility that a single transaction is recorded in multiple local blocks. The traditional solution, which compares each transaction across all shards, results in significant computational and time overhead, making it an impractical approach for large-scale networks. The DAG-based approach is posited as an alternative

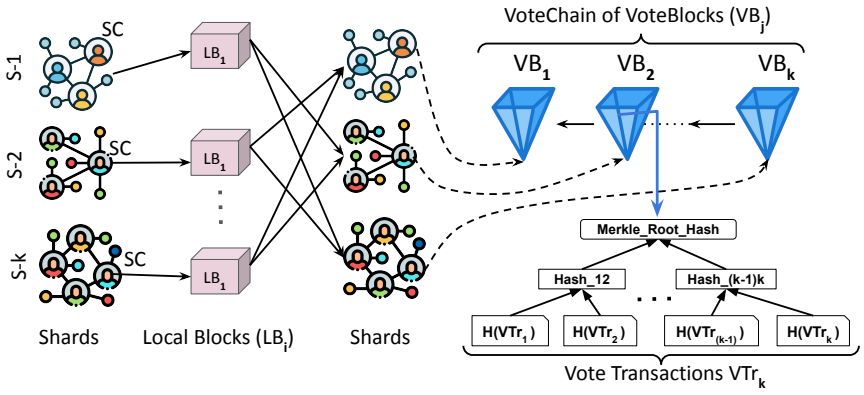


Figure 3.5: This figure illustrates the generation of local blocks (LB_i) and VoteChain of VoteBlocks (VB_j). A Shard Coordinator (SC) is elected in each shard, as depicted here.

to traditional methods for forming global blocks. DAG allows for a nonlinear arrangement of transactions. For each transaction T_{ij} in a local block B_i, a node is created in the DAG. If T_{ij} references another transaction T_{mn} in a different local block B_m, a directed edge is drawn from T_{ij} to T_{mn}. If two transactions, T_{ij} and T_{xy}, reference the same input transaction T_{mn} (i.e., if $T_{ij} \rightarrow T_{mn}$ and $T_{xy} \rightarrow T_{mn}$ exist in the DAG), it indicates potential double-spending. The system flags T_{ij} and T_{xy} for further validation. Each flagged transaction is then individually validated. The validation of each transaction is done by checking timestamp(T_{ij}) and timestamp(T_{xy}). If timestamp$(T_{ij}) <$ timestamp(T_{xy}), then T_{ij} is considered valid, and T_{xy} is marked as invalid (double spend), and vice versa. If timestamp(T_{ij}) and timestamp(T_{mn}) have the same timestamp, secondary criteria such as transaction ID and nonce value are used to determine the original transaction. The transaction with the lower lexicographical order or lower nonce value is considered the original. Additionally, if T_{ij} has a lower timestamp than T_{mn} but is broadcasted by an adversary much later than T_{mn}, nodes track the broadcast time and validate transactions within a predefined time window to ensure timely detection of adversarial delays. Initially, T_{mn} would be accepted due to its earlier broadcast time. If T_{ij} is later detected within the allowable time window and validated as the original transaction based on the lower timestamp, T_{mn} would be marked as invalid.

Once the original (valid) transaction is identified, the conflicting (invalid) transaction is discarded or marked as a double spend. The DAG is then updated after the invalid transaction is removed. This update might propagate through the DAG, affecting how other transactions are connected. The updated status on valid and invalid transactions is broadcasted to the entire network, ensuring that all nodes and shards have the latest and accurate state of the ledger. The global block is finalized with the remaining valid transactions and committed to the global ledger. Efficient data structures (i.e., DAG) and the sharding mechanism ensure that the global block remains practical in size. By assuming a maximum of 40-45 shards, each handling 500-800 transactions per local block, UnoShard can manage a global block size of up to 36,000 transactions. This approach leverages proven methodologies from high-throughput blockchains like EOS [140] and Solana [141], ensuring an effective and scalable block size even when aggregating transactions from multiple shards.

The *Algorithm* 6 describes the process of compiling transactions from local blocks of individual shards into a unified global block. It starts by constructing a DAG where each transaction from the shards acts as one of the nodes of DAG construction. This DAG reflects the dependencies between transactions on different shards. The algorithm then identifies possible double-spending situations by looking for transactions

referencing the same input in the DAG. These suspect transactions are validated based on their timestamps, determining which transaction came first and, therefore, is valid. Transactions found to be double spends are marked as invalid and removed from the DAG. After resolving these conflicts, the algorithm compiles all valid transactions into a global block, which is then broadcasted for network-wide verification and subsequently committed to the global ledger. This approach efficiently maintains the integrity and consistency of the blockchain ledger in a sharded architecture.

Algorithm 6: Global Block Generation

Input: LocalBlocks (Array of local blocks from each shard),
DAG (Directed Acyclic Graph representing transactions)
Output: GlobalBlock (Finalized global block for the ledger)
Function `globalBlock(`*block, nonce*`)`**:**

```
Initialize an empty DAG;
for Every LB_i in LocalBlocks do
    for Every T_ij in LB_i do
        Create a node in the DAG for T_ij;
        if T_ij references another T_mn in a different LB_m then
            Draw a directed edge from T_ij to T_mn in the DAG;
for Every ((T_ij, T_mn)) in the DAG that reference the same input transaction do
    Flag (T_ij, T_mn) for validation;
    Validate each flagged transaction by comparing timestamps;
    if (T_ij == T_mn) then
        Compare Transaction IDs;
        if (T_ij < T_mn) then
            Mark T_ij as valid and T_mn as invalid;
        else
            Mark T_mn as valid and T_ij as invalid;
    else
        if (T_ij < T_mn) then
            Mark T_ij as valid and T_mn as invalid;
        else
            Mark T_mn as valid and T_ij as invalid;
    Update the DAG by removing links associated with invalid transactions;
Broadcast(Tr_update) ;
create(GlobalBlock) ;
commit(GlobalBlock)
```

3.6 PERFORMANCE ANALYSIS

The performance analysis of BlockTree is performed based on four parameters: consensus delay, throughput, fairness, and block congestion control. We found these four parameters are very crucial for the perfor-

mance of the blockchain. First, we provide the simulation environment of BlockTree, which includes some important assumptions about the simulation.

3.6.1 Simulation Environment

We evaluate BlockTree with 1000-node experiments on a customized simulation with parameters like transaction frequency, consensus delay, fairness, and leader selection time based on different block sizes (512 KB to 9MB). Our simulation assumes a network delay of 100ms with a variance of 10ms to reflect typical conditions in a large-scale distributed blockchain network. We also assume a message loss rate of 0.1% to represent a reliable network environment and utilize a mesh network topology, where each node is connected to multiple other nodes, ensuring robustness and resilience. We compared BlockTree with Bitcoin, Bitcoin Cash, Bitcoin-NG, Algorand, Obscuro, and Prism. In our analysis, we referenced the empirical performance data presented in the original studies of state-of-the-art protocols, enabling a consistent and fair evaluation against BlockTree. We endeavored to conduct the most thorough comparative analysis possible, given the resources available, ensuring that our evaluation was aligned with established standards in the field. The evaluation results show that it is possible to improve consensus delay and throughput (number of transaction executions) by modifying the structure and consensus algorithm while maintaining the same level of security. BlockTree outperforms almost all the metrics across all the measured parameters. Fork generation, a common drawback in most previous cryptocurrencies, has been addressed in the BlockTree protocol. We run the protocol with different block sizes to test throughput, scalability, consensus delay, fairness, and block congestion control, as seen in the comparative graphs below. We vary the block sizes from 512 KB to 9MB. It takes more time to verify and propagate as the block size increases.

3.6.2 Consensus Delay

Figure 3.6a shows the consensus delay with varying block sizes. The results indicate that as the block size increases, the consensus delay also increases because a larger block takes more time to verify and propagate. However, in BlockTree, the consensus delay is significantly lower than that of state-of-the-art schemes such as Algorand, Bitcoin-NG, and Prism. This demonstrates that BlockTree can perform transactions within a few seconds, even as block sizes grow. One of the key advantages of BlockTree is its m-way tree structure, which allows for parallel processing and efficient transaction management. This structure helps maintain lower consensus delays as the network scales. During our tests, we found that the m-way nature of BlockTree ensures that the consensus delay remains minimally affected as the number of miners increases.

When the number of miners surpasses the defined limit m, the network automatically splits and creates a new shard. This dynamic shard creation ensures that BlockTree does not face scalability issues that typically plague other blockchain protocols. Additionally, the efficient leader selection algorithm and the streamlined verification process further contribute to maintaining low consensus delays, even with increasing network size. As a result, BlockTree outperforms other schemes not only in scalability but also in maintaining rapid transaction processing times, making it a robust solution for high-throughput blockchain applications.

3.6.3 Throughput

Figure 3.6b shows the transaction frequency with varying block sizes. The results indicate that BlockTree performs exceptionally well compared to other schemes. As the block size increases, the transaction frequency also increases, thereby enhancing the throughput of BlockTree. This superior performance can be attributed to the m-way tree structure of BlockTree, which allows for efficient parallel processing and management of transactions across multiple shards. Additionally, we tested the throughput of BlockTree with varying time intervals between leader nodes, ranging from 30 seconds to 100 seconds. The results show that as the time interval between leader nodes increases, the throughput also increases. However, it is important to note that if the interval increases excessively, it may lead to a situation similar to monologues, where a single leader node generates an excessive number of data blocks. Therefore, the interval should not exceed a reasonable limit to ensure fairness and maintain optimal performance.

Unlike other state-of-the-art schemes like Algorand, Bitcoin-NG, and Prism, BlockTree consistently achieves higher throughput. This is due to its innovative consensus mechanism and efficient block verification process. The ability to dynamically create new shards as the network grows further ensures that BlockTree can handle a significant increase in transaction volume without compromising performance. As a result, BlockTree improves scalability and maintains high throughput, making it a suitable solution for large-scale blockchain applications. The throughput analysis clearly demonstrates that BlockTree can handle a high volume of transactions efficiently, even with increasing block sizes and varying leader selection intervals. This capability makes BlockTree a robust and scalable blockchain protocol suitable for diverse and demanding applications.

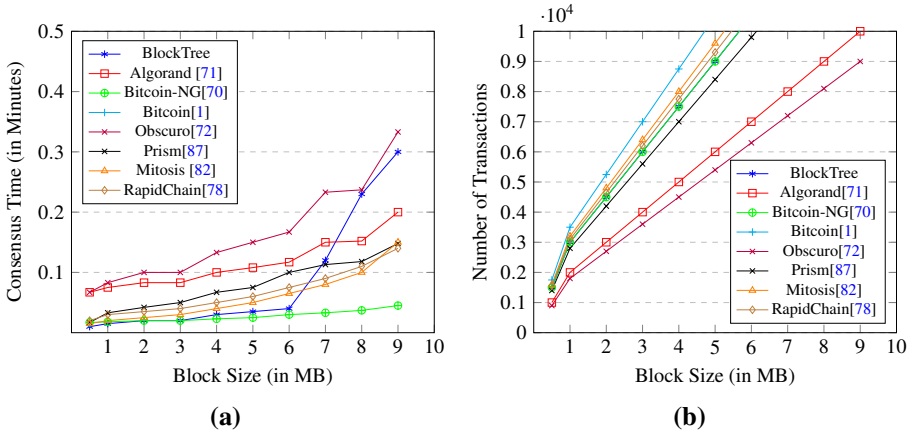


Figure 3.6: Comparative analysis of BlockTree with state-of-the-art schemes based on the (a) consensus delay and (b) throughput

3.6.4 Fairness

Figure 3.7a illustrates the fairness of block proposers in BlockTree compared to other state-of-the-art schemes. As the number of miners increases, the likelihood of different miners generating blocks decreases in many traditional schemes. However, BlockTree demonstrates superior fairness across varying numbers of miner nodes, from 100 to 1000, maintaining a more equitable distribution of block generation opportunities. This enhanced fairness in BlockTree is primarily due to its efficient sharding mechanism. BlockTree

maintains a fixed, defined size for each shard, ensuring that every node within a shard has an equal opportunity to become a leader node. This approach contrasts with other schemes where the competition among all miners can lead to disproportionate chances for block generation, often favoring more powerful nodes. In BlockTree, as the number of miners increases and surpasses the defined limit for a shard, the network splits, and a new shard is created. This dynamic creation of shards helps distribute the mining power more evenly across the network, preventing any single node or a small group of nodes from dominating the block generation process. The fixed shard size ensures that each shard remains manageable and that the nodes within each shard operate under fair conditions.

Unlike Algorand, Bitcoin-NG, and Prism, BlockTree's sharding mechanism provides a significant advantage in maintaining fairness. By ensuring that each node has a uniform chance to become a leader, BlockTree mitigates the risk of centralization and promotes a more decentralized and equitable blockchain environment. This makes BlockTree fairer but more resilient and secure, as it prevents the concentration of mining power that could otherwise lead to potential vulnerabilities and attacks. Overall, the fairness analysis underscores BlockTree's capability to provide a balanced and just platform for block generation, even as the network scales up, reinforcing its position as a leading blockchain protocol regarding fairness and decentralization.

3.6.5 Block Congestion Control

In cryptocurrencies like Bitcoin, Bitcoin-NG, Algorand, and others, every miner competes with the rest of the nodes to win block creation compensation. This competition often leads to block congestion as many nodes generate blocks simultaneously, resulting in illegitimate fork generation. BlockTree effectively addresses this issue by creating multiple fixed-size shards. Instead of having all miners compete for the same block, these shards allow different groups to create different blocks for separate transactions. This approach significantly reduces block congestion and improves throughput. BlockTree's sharding mechanism ensures that each shard operates independently, minimizing the competition within a shard and distributing the workload evenly across the network. As a result, the likelihood of block congestion and forks is greatly reduced.

Figure 3.7b illustrates the block congestion control capacity of BlockTree compared to state-of-the-art schemes. The results indicate that BlockTree outperforms other schemes by maintaining lower levels of congestion and higher throughput. This performance improvement is primarily due to the efficient sharding mechanism, which distributes transactions across multiple shards, preventing bottlenecks and ensuring a smoother transaction flow. Additionally, the fixed-size shard structure in BlockTree ensures that each shard remains manageable and operates under optimal conditions. This prevents any single shard from becoming overwhelmed with transactions, further contributing to the overall efficiency and scalability of the network. By reducing congestion, BlockTree not only enhances throughput but also improves the overall user experience by ensuring faster transaction confirmations and reducing the risk of network slowdowns. Consequently, BlockTree's approach to block congestion control through sharding provides a significant advantage over traditional blockchain protocols. It effectively mitigates the issues of block congestion and fork generation, leading to a more robust, scalable, and efficient blockchain network.

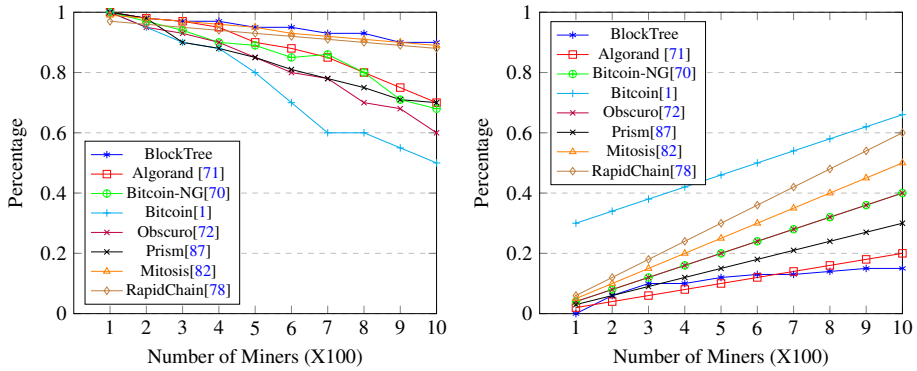


Figure 3.7: Comparative analysis of BlockTree with state-of-the-art schemes based on (a) fairness and (b) block congestion control

3.7 SECURITY ANALYSIS

We have analyzed the security of BlockTree based on three parameters: double-spending, illegitimate fork, and scalability. Below is the qualitative discussion on how BlockTree has prevented these attacks and maintains high scalability.

Proposition 1. *In the UnoShard system, the probability $P_{takeover}$ of 1% attack or shard takeover attack, defined as more than half of the nodes in any shard being malicious, is negligibly small.*

Proof. Let N Total number of nodes in the network, S: Number of shards in the network, $N_S = \frac{N}{S}$: Number of nodes per shard, assuming an even distribution, T: Threshold for shard takeover, defined as the majority ($> \frac{N_S}{2}$) of malicious nodes in a shard, and P_M: Probability that any given node is malicious. The probability of a shard takeover in the UnoShard blockchain network can be modeled as a binomial probability problem. We calculate the probability that more than half of the nodes in a shard are malicious. The binomial distribution gives the probability of exactly k nodes in a shard being malicious:

$$P(X = k) = \binom{N_S}{k} P_M^k (1 - P_M)^{N_S - k} \tag{3.10}$$

To find the probability of a shard takeover, we need to calculate the cumulative probability of more than half the nodes in a shard being malicious. This is given by:

$$P_{takeover} = \sum_{k=\frac{N_S}{2}+1}^{N_S} \binom{N_S}{k} P_M^k (1 - P_M)^{N_S - k} \tag{3.11}$$

where the sum accumulates the probabilities for all scenarios where more than half of the nodes are malicious. However, in all the scenarios affecting the value of $P_{takeover}$, the P_M is the most vulnerable parameter because, in a public blockchain, where any node can join and participate in the consensus process with the ware minimum authentication process, it is hard to say P_M could be typically low. Apart from the P_M, the other factors are fair shard allocation based on DPoS and trust-based intra-shard consensus.

In UnoShard, the shard allocation process is governed by DPoS, where only a selected group of nodes (delegates) participate in consensus. These delegates are elected based on their reputation or stake, which reduces the likelihood of malicious nodes gaining significant influence, thereby effectively lowering P_M over time. Furthermore, The fair shard allocation mechanism ensures an even distribution of nodes across shards, making it difficult for any single shard to be dominated by malicious nodes. Even if the initial P_M is higher, the distribution helps keep $P_{takeover}$ low. Along with fair shard allocation using DPoS, the trust-based consensus mechanism, which dynamically adjusts the influence of nodes based on their behavior, plays a big role in lowering the probability of shard takeover. Malicious nodes quickly lose trust, limiting their impact on the network's decision-making process. This mechanism acts as a secondary layer of security, ensuring that even if P_M is higher initially, its influence on shard takeover risk is mitigated. Given these mechanisms, the cumulative effect is that $P_{takeover}$ remains low, though it may not be "negligibly small" in the strictest mathematical sense. The assertion of "negligible" should be understood in the context of these mitigating factors rather than as an absolute statement. □

Proposition 2. *The UnoShard integrates a fair shard allocation method, trust-based intra-shard consensus, and cross-shard consensus with the DAG to secure from majority attack. This is defined by the system's ability to manage node failures or malicious activity while maintaining the integrity and operation of the whole network.*

Proof. UnoShard's resilience against majority attacks is achieved through the integration of fair shard allocation, a trust-based Practical Byzantine Fault Tolerance (PBFT) consensus mechanism within each shard, and a cross-shard consensus process that utilizes a Directed Acyclic Graph (DAG). This combination of mechanisms ensures that UnoShard can handle node failures or malicious activities while maintaining the network's integrity and operational continuity.

Initially, the UnoShard begins by evenly distributing nodes across multiple shards. Let S represent the number of shards and N the total number of nodes, resulting in each shard containing $N_S = \frac{N}{S}$ nodes. This fair shard allocation minimizes the likelihood of any single shard being disproportionately controlled by malicious nodes, as it prevents the concentration of such nodes within one shard. In the next phase, the UnoShard employs the PBFT consensus mechanism within each shard, which requires at least $\frac{2}{3}$ of the nodes to agree for a transaction to be validated. This consensus model allows the system to tolerate up to $\frac{1}{3}$ of the nodes being faulty or malicious. The resilience of each shard against majority attacks is represented by $R = 1 - \frac{1}{3N_S}$ indicating that as long as fewer than one-third of the nodes in a shard are compromised, the shard will remain secure and operational. Lastly, the cross-shard consensus process in UnoShard leverages a DAG to manage and validate transactions across shards, ensuring consistency and resolving any conflicts that arise from interactions between shards. The use of DAG enhances the fault tolerance of the entire network by synchronizing the state of all shards, thereby maintaining the overall network's integrity even if some shards encounter issues.

The combined fault tolerance of UnoShard can be expressed as $P_{\text{tolerance}} = P_f \times R$ where P_f is the probability that a single shard failure impacts the network. Given that P_f is proportional to $\frac{1}{S}$, the likelihood of a shard failure affecting the entire network decreases as the number of shards increases. This, combined with the shard-level resilience R, ensures that UnoShard is robust against majority attacks, providing a secure and reliable blockchain network even in the presence of malicious nodes. □

3.7.1 Prevention of Double Spending

BlockTree: Double spending means spending the same money twice. It is not an issue in physical currency, but it is a big problem in digital currency. In digital currency, a transaction carrying the digital cash is broadcasted to all the nodes in the network for verification. It is possible that another node copies the transaction. Before confirmation, it was rebroadcast into the network. Double spending attacks are a big concern for Bitcoin, Bitcoin-NG, and other cryptocurrencies. In the case of Bitcoin, Bitcoin-NG, this issue remains there due to the long confirmation time. However, in BlockTree, all the transactions are bunched in the data block via the leader node only. While the leader node generates the data block, no other node can create the data block. Therefore, the possibility of rebroadcasting the same transaction again in the network is eliminated. Hence, the possibilities of double-spending are negligible.

Proposition 3 (UnoShard)**.** *The cross-shard consensus in UnoShard uses a Directed Acyclic Graph (DAG) to record transactions in a global block. Let T_{ij} and T_{xy} be transactions in UnoShard which references the same input transaction T_{mn}. The DAG structure ensures that UnoShard validates at most one of T_{ij} and T_{xy}, thereby preventing double spending attacks.*

Proof. Each transaction in the DAG model of UnoShard is a vertex. When two vertices T_{ij} and T_{xy} relate to a common input T_{mn}, directed edges towards T_{mn} are produced, implying the possibility of double-spending. A time-based validation is the solution: If $Timestamp(T_{ij}) < Timestamp(T_{xy})$, then T_{ij} is considered as valid, while T_{xy} is considered as invalid; otherwise, vice versa. The DAG then undergoes pruning to remove invalid transactions, denoted as $\mathscr{F} : DAG \rightarrow DAG'$, where DAG' is the updated graph. Valid transactions from DAG' are aggregated into the global block, denoted as $\mathscr{G} : DAG' \rightarrow GB$, where GB represents the finalized global block. This mechanism guarantees that only transactions without double-spending issues are included, preserving the blockchain system's integrity. □

3.7.2 Illegitimate Fork Prevention

In the BlockTree protocol, several mechanisms ensure that the leader cannot create forks and that any attempts to do so are effectively managed. The leader selection algorithm guarantees that only one leader is active at any given time. This selection process involves solving a cryptographic puzzle and incorporates checks on the status and age of nodes. Nodes with a "bad" status are excluded, and among the remaining nodes, the one with the lowest age is preferred, ensuring fair and reliable leader selection. The selected leader's identity is broadcast to the network, ensuring transparency. BlockTree distinguishes between leader blocks and data blocks. Once a leader is selected, a leader block is first created, marking the beginning of the term. During the specified epoch, the leader generates data blocks. This distinction helps maintain order and prevent unauthorized block creation. This concept is similar to the Bitcoin-NG protocol, where the blockchain is divided into epochs managed by a leader who generates key blocks and microblocks. Key blocks establish the leader for an epoch, while microblocks contain transactions. This separation ensures efficient transaction processing and prevents issues like stale blocks common in traditional blockchain protocols. Unlike Bitcoin-NG, where the current leader node continues generating microblocks until a new key block arrives, BlockTree restricts the leader's activity to a specific time interval. After this interval, all nodes switch to leader selection mode to choose a new leader. This approach prevents the problem of unnecessary fork generation, ensuring no overlap in leader activities. Thus, the possibility of parallel data block generation by multiple nodes in the same group is minimized, effectively eliminating forks within the same group. The BlockTree consensus algorithm follows the 2K+1 rule, ensuring that the number of honest

nodes is always greater than the number of malicious nodes. For a block to be accepted, it must receive votes from at least 2K+1 nodes.

If a leader attempts to create a fork by generating two data blocks with the same parent, the protocol's verification processes detect this. Nodes continuously verify blocks, and any discrepancies in hash values indicate a fork attempt. Honest nodes will vote only for the valid block, rejecting the conflicting one. The block with the highest number of valid confirmations is added to the main chain, and the conflicting block is discarded. Malicious leaders are penalized by being marked with bad status, excluding them from future leader selections.

3.7.3 Scalability Enhancement

Apart from the fork and double-spending, the high transaction load is also a limitation of blockchain implementation, where every transaction is validated and executed on every node. To maintain security and improve scalability, public blockchains impose upper limits that manage the rate at which they can process transactions and the rate at which new blocks are created. In Bitcoin and Bitcoin-NG, as they are public blockchains, there might be any number of nodes who race to mine the block as soon as possible to get rewarded. Due to this race amongst nodes, much of the mining power goes in vain. Also, this leads to the scalability issue. To deal with the scalability issue and utilize the mining power, BlockTree modified the blockchain structure into the m-way tree-like structure. BlockTree creates the shards when the number of miners becomes greater than the defined size m, and a new fork is created in the current chain from that instant. This fork is the legitimate fork in the chain. A newly created shard starts mining in the new fork of the chain. In this way, the mining competition is reduced, and now miners in all the shards mine the blocks parallelly. For example, if 1000 miners were mining the same set of transactions and were now divided into ten shards of 100 miners, the blockchain's throughput would increase by ten times.

Concluding Remark

In this chapter, we have addressed the critical issue of scalability in public blockchains by introducing two novel consensus algorithms: BlockTree and UnoShard. Through an in-depth exploration of their architectures and mechanisms, we have demonstrated how these protocols significantly enhance the scalability and throughput of blockchain networks while maintaining high levels of security. BlockTree's innovative structure allows for efficient block validation and propagation, mitigating common bottlenecks that hinder scalability. Similarly, UnoShard distributes the workload across multiple shards, enabling the system to scale effectively without compromising security. UnoShard ensures fair shard allocation and fair load distribution, along with the prevention of shard takeover attacks and double spending attacks. Our comparative analysis reveals that both BlockTree and UnoShard outperform state-of-the-art scalability solutions, offering promising directions for the future of scalable public blockchains. Building on the advancements made in this chapter, we now turn our focus to the challenge of interoperability in blockchain networks in Chapter 4. As blockchain technology continues to evolve, the ability to seamlessly communicate and transact across different blockchain platforms has become increasingly important. Chapter 4 introduces the *CrossLedger* protocol, designed to enable secure and efficient cross-chain asset transfers. *CrossLedger* paves the way for a more interconnected and versatile blockchain ecosystem by addressing the complexities of interoperability. This transition from scalability to interoperability reflects the broader goal of enhancing blockchain technology to meet the diverse needs of its users across different platforms.

4

Interoperability

The development of real-world blockchain applications based on different platforms (e.g., Bitcoin, Ethereum, Hyperledger Fabric, etc.) shows that blockchain technology is maturing at a fast pace. Blockchain adaptability is a motivating factor for the existence of different blockchains, leading to a heterogeneous ecosystem [92]. With the existence of different blockchains, some questions arose that need to be answered, such as: 1. What if the particular service running on the blockchain becomes vulnerable or is shut down? 2. If user requirements change for which a different blockchain is more appropriate, then how to handle this condition? and 3. What if one blockchain user wants to transfer something (asset/data) to another user of a different blockchain? All these questions can be answered if blockchain technology is enabled with the interoperability feature [28]. Blockchain interoperability can be defined in a simple term as the ability of one blockchain (source blockchain) to change the state of another blockchain (destination blockchain) enabled by cross-chain transactions. A technical report from the National Institute of Standards and Technology (NIST) defines blockchain interoperability as [142]

"*A composition of distinguishable blockchain systems, each representing a unique distributed data ledger, where atomic transaction execution may span multiple heterogeneous blockchain systems, and where data recorded in one blockchain are reachable, verifiable, and referable by another possibly foreign transaction in a semantically compatible manner.*"

There are three categories of blockchain interoperability [96]: first, Interoperability between different blockchains. Second, interoperability between decentralized applications using the same blockchain, and third, interoperability blockchain with technologies (i.e., integration with enterprise systems). In this work, we mainly focused on the first category of interoperability, i.e., interoperability between different blockchains. Therefore, from now on, when we say interoperability, we are referring to the first category of interoperability. While examining the state-of-the-art techniques, we identify the following challenges in existing interoperability solutions:

4.1 CHALLENGES IN INTEROPERABLE BLOCKCHAINS

Interoperability between different blockchain platforms remains one of the most significant hurdles the blockchain faces today. As blockchain technology continues to evolve and find applications across various sectors, the ability for different blockchain networks to seamlessly interact with one another is becoming increasingly crucial. However, achieving this interoperability is fraught with many challenges. These obsta-

cles come from differences in blockchain architectures, consensus mechanisms, the absence of standardized protocols, third-party intermediaries, and the need to satisfy the five essential characteristics that complicate the integration of diverse blockchain systems together.

1. **Different Blockchain Architectures:** Blockchains come in various architectures, each with specific protocols, consensus mechanisms, and security frameworks. For example, Bitcoin and Ethereum substantially differ in structure and function, complicating direct interchain interactions.
2. **Diverse Consensus Mechanisms:** Blockchains operate on varied consensus mechanisms. While Bitcoin uses PoW, others like Cardano use PoS. As the backbone of transaction verification, these mechanisms pose challenges when blockchains try to converse.
3. **Standards and Protocols:** The absence of standardized protocols complicates the development of solutions that enable seamless interaction between different blockchain systems, requiring significant effort and coordination to create interoperable frameworks.
4. **Third Party Intermediary:** Many current solutions for blockchain interoperability rely on third-party intermediaries. This requirement introduces potential vulnerabilities and deviates from the decentralized essence of blockchain technology, making systems more susceptible to attacks or failures centralized around these intermediaries.
5. **Presence of Five Essential Characteristics:** For accomplishing an efficient cross-blockchain transfer, blockchain interoperability must pose the five essential characteristics: non-repudiation, unlinkability, confidentiality, atomicity, and interoperability. The existing solutions have some characteristics, but not all of them.

4.2 *CROSSLEDGER*: A PIONEER CROSS-CHAIN ASSET TRANSFER PROTOCOL

We have proposed *CrossLedger*, a novel cross-chain asset transfer protocol to enable blockchain interoperability while addressing the above-mentioned challenges. It's important to note that, unlike the state-of-the-art approaches, the *CrossLedger* is a heterogeneous protocol, meaning that the source and destination blockchains for asset transfers are not fixed. Furthermore, *CrossLedger* also eliminates the necessity of a third-party intermediary. In contrast to state-of-the-art methods, which specify the source blockchain, *CrossLedger* allows for the use of multiple blockchains on both ends. For each blockchain, *CrossLedger* employs a novel Asset Forwarder Selection (AFS) Algorithm 7 to determine which Asset Forwarder (AF) forwards assets from the source blockchain to the destination blockchain. Between the asset forwarders on both blockchains, *CrossLedger* used the Trust Establishment (TE) Algorithm 8. Furthermore, the Asset Transfer and Confirmation (ATC) Algorithm 9 ensures that the asset is transferred successfully between blockchains. When compared to state-of-the-art approaches, all three of these algorithms maintain consistency across blockchains to give high levels of security, throughput, and scalability. The contributions of *CrossLedger* are as follows:

- We have proposed a generalized Cross-chain asset transfer protocol known as *CrossLedger* for two different types of blockchain platforms. Additionally, *CrossLedger* eliminates the third-party inter-

mediary using the AFS and TE algorithms. As a result, the proposed protocol is more secure and reliable.

- *CrossLedger* introduces a novel Asset Forwarder Selection (AFS) algorithm to select an Asset Forwarder (AF) in each blockchain, i.e., source and destination blockchains. The AF is responsible for transferring the asset from the source blockchain to the destination blockchain. *CrossLedger* uses a novel Trust Establishment algorithm between the AFs of both the blockchains. A novel Asset Transfer and Confirmation (ATC) algorithm ensures that the asset is transferred successfully between blockchains.
- The proposed *CrossLedger* structure supports non-repudiation, unlinkability, confidentiality, atomicity, and interoperability characteristics for Cross-chain based asset transfer.
- We have provided the security analysis of the *CrossLedger*, which illustrates that the *CrossLedger* is secured against the double-spending attack, liveness attack, and Sybil attack.

4.2.1 System Model

This section describes the system model of our proposed scheme. Figure 4.1 depicts the typical process of asset transfer from user U_A to user U_B. There are two blockchains, blockchain A and blockchain B. These blockchains are of any type, i.e., public-private, public-public, and private-private. In other words, blockchain A can be Bitcoin type, and blockchain B can be Ethereum type or Hyperledger fabric type. The *CrossLedger* is a generalized asset transfer protocol where the source and destination blockchains are not fixed. The numbers from 1 to 10 shown in the figure are the sequence of the operations for the asset transfer. Apart from the users U_A and U_B, the other elements of the system are miners and Asset Forwarders AF_A, AF_B. The highlighted blocks B_3 and B_5 are the source and destination blocks, respectively, showing that the asset(s) is/are included in those blocks. When the asset is successfully transferred, the AF_B is responsible for confirming the asset to both the users U_A, U_B. Below, we have defined the system model's main components: User, Asset, Asset Forwarder, and Asset Transfer Confirmation.

1. **User:** A User denoted as U_A/U_B is the one who generates transactions. The actual asset transfer happens between the users, where one user acts as the sender, and another acts as the receiver of the asset. In Figure 4.1, user U_A is the sender, and user U_B is the receiver that belongs to two different blockchains, blockchain A and blockchain B.
2. **Asset:** An Asset *Ast* is a digital representation of arbitrary data, fungible assets, or non-fungible assets. Arbitrary data is often represented via a key-value pair, which is a suitable representation of some blockchains. The fungible assets are usually the payment tokens, and non-fungible tokens are utility tokens such as ERC20 tokens.
3. **Asset Forwarder (AF):** A AF is one of the nodes selected from the distributed network based on the consensus algorithm executed in the blockchain network. The AF is responsible for the asset transfer and confirmation from the source blockchain to the destination blockchain. An AF is selected on both the blockchain, i.e., source blockchain A and destination blockchain B.
4. **Asset Transfer and Confirmation (ATC):** The ATC algorithm verifies the successful transfer of the asset *Ast* from the source blockchain to the destination blockchain. In practice, rolling back or modifying the asset *Ast* after confirmation is computationally difficult.

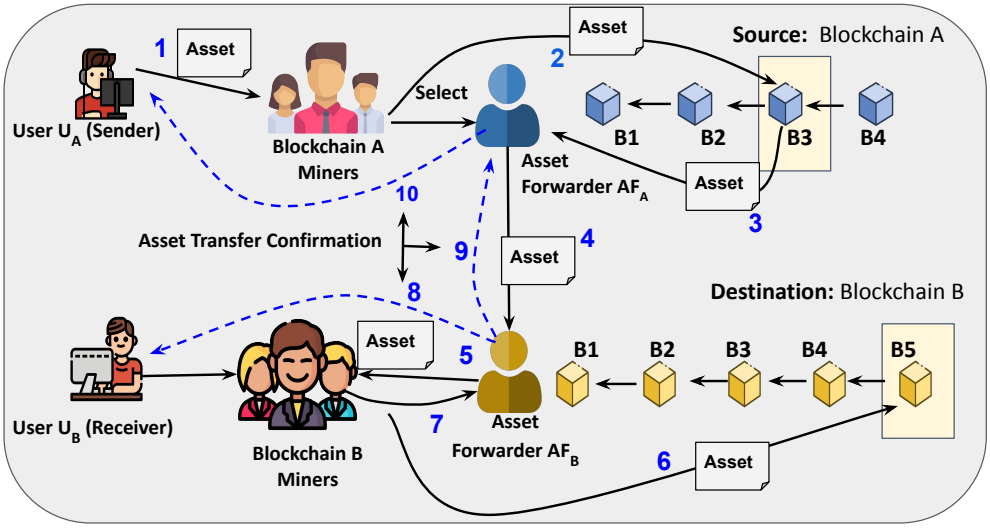


Figure 4.1: *CrossLedger*'s asset transfer mechanism between two blockchains, source blockchain A and destination blockchain B

The *ATC* certifies that the asset *Ast* is successfully committed from the source blockchain to the destination blockchain. In effect, it is impossible to computationally revert or alter the asset *Ast* after the confirmation. In *CrossLedger*, *ATC* is deterministic. When a *Ast* is included in a block and appended to the blockchain, it is termed deterministic confirmation. This is accomplished through the proposal of the block by an elected *Node* with adequate authority. In this instance, an asset forwarder is chosen and accountable for asset transfer and confirmation.

4.2.2 Background

This section briefly describes the taxonomy of the *CrossLedger* scheme, which includes the basic definition of Ownership and the adversary model. In the adversary model, we have defined some assumptions that make it possible for the adversary to attack the system. We have also proved in the later section how the adversary cannot make an attack.

A. Five Essential Properties for Cross-Blockchain Transfer

In CrossLedger, we have considered five essential properties that are crucial for a solution of cross-chain asset transfer. The five properties are as follows: Non-Repudiation, Unlinkability, Confidentiality, Atomicity, and Interoperability. Below, we define these properties and why we consider them essential while designing CrossLedger.

1. **Non-Repudiation:** This property ensures that once a transaction has been completed, i.e., sent by the sender and received by the receiver, neither the sender nor the receiver can deny their involvement in the transaction, which means it provides irrefutable proof of the origin and reception of the asset transfer. This property becomes highly important in a decentralized environment where there is no central authority to enforce the contracts. The proposed CrossLedger is also a protocol to transfer assets across the different blockchains without the involvement of a third party. Therefore, keeping the

non-repudiation property is highly important. Non-repudiation ensures that once an asset is recorded, both chains are accountable for their actions.

2. **Unlinkability:** This property ensures that transactions or assets transferred across blockchains cannot be linked back to their origin or to each other, thus protecting the privacy of users. In cross-blockchain environments, preserving user privacy is a significant concern. Unlinkability prevents adversaries from tracing transactions across different blockchains, which could reveal sensitive information about user behavior or asset ownership. This is particularly important when dealing with public blockchains, where transaction data is visible to all participants.

3. **Confidentiality:** This is an important security property for any digitally secured system, especially when the system includes some sensitive information in the transactions. Confidentiality ensures that the details of the assets or transaction, such as the value of the asset, sender, and recipient, are not accessible to unauthorized parties. Confidentiality is equally essential in cross-chain transfers to protect sensitive information from being exposed during the transfer process. As transactions move between different blockchains, which may have varying levels of security and transparency, ensuring that only the intended parties can view the transaction details is crucial for maintaining trust and security.

4. **Atomicity:** The concept of atomicity is drawn from the database system, which ensures that either the transactions are fully completed or not executed at all. This means that if an asset is transferred from one blockchain to another, the transfer must be entirely successful on both blockchains, or the transaction should be rolled back. Atomicity is vital to prevent partial transfers that could lead to asset loss or inconsistencies between blockchains. In a cross-chain environment, where different blockchains might have different rules and processing times, atomicity ensures that assets are not left in an incomplete state, thereby preserving the integrity of the transfer process.

5. **Interoperability:** This is the core essential property for any cross-chain transfer protocol. Any protocol is said to be interoperable if it ensures the ability of different blockchain networks to work together, allowing assets, data, and transactions to move seamlessly between them without requiring a trusted intermediary. The intent of designing the *CrossLedger* is also the same: there should not be any intermediary or third party to do the task of transferring from one blockchain to another blockchain. Without interoperability, cross-chain transactions would not be possible, as each blockchain would function in isolation. Interoperability ensures that different blockchains, regardless of their underlying technology, can communicate and process transactions in a coordinated manner.

B. Ownership

The ownership of the asset is defined using the following definition.

Definition 5 (Ownership). *The ATC can only be successfully executed if the following conditions are satisfied.*

1. *User U_A (sender) owns the Asset_Value.*
2. *User U_A (sender), U_B (receiver) should be such that ($(U_A, U_B) \in \bigcup_{i=1}^{n} U$), where U is the set of all the legitimate users in both the blockchains.*
3. *Asset_Value should belongs to the legitimate user ($\in U$) and Asset_Value $\in \bigcup_{i=1}^{m} Asset$.*

4. *An asset has exactly one owner at a time* $Owner(U_A, Asset_Value)$ *is true if and only if* U_A *owns* $Asset_Value$.

C. Asset Evaluator

The present value of assets is calculated using book values, fundamental valuation methodologies such as discounted cash flow analyses, option pricing models, or comparables. The following is the definition of the asset evaluator function in terms of $CrossLedger$.

- It begins by verifying that the asset forwarder has enough amount of the asset. This step is known as the balance function, and it produces the following result: If the asset forwarder has sufficient units of the asset, then the value is true; otherwise, the value is false.
- If the prior transaction references the current transaction, it is checked to see if the previous transaction is part of a valid block.

If both of the above steps are true, the asset Verifier function returns true; otherwise, it returns false. After completing all transactions, they return a receipt with all the transaction details.

D. Traceable Ring Signature (TRS)

Unlike ring signatures, *TRS* limits "excessive" anonymity [143]. A *TRS* is a tuple of four elements as $\sum_{TRS} = (\mathbb{G}, \mathbb{S}, \mathbb{V}, \mathbb{T})$. We consider the ordered set of public key $\mathbb{S}|_{pk} = (pk_{N_1}, ..., pk_{N_n})$ as a ring. The four tuples of the *TRS* can be defined as follows.

1. **Generator ($\mathbb{G}$):** It is a function that takes the security parameter, let's say, k as input and generates the output public-secret key-pair (pk_{N_i}, sk_{N_i}).

$$(pk_{N_i}, sk_{N_i}) \leftarrow \mathbb{G}(k) \tag{4.1}$$

2. **Signature ($\mathbb{S}$):** It is a function that takes three parameters as input, which is the secret key sk_{N_i}, tag field $Tg = (issue, pk_{N_n})$, and message Msg, and generate the outputs as signature σ_{Msg}.

$$\sigma_{Msg} \leftarrow \mathbb{S}(sk_{N_i}, Tg, Msg) \tag{4.2}$$

3. **Verifier ($\mathbb{V}$):** It is the function of signature verification that takes the tag field Tg, message Msg and signature σ_{Msg}, as the input and generates the one-bit boolean output as 0/1, indicate the valid or invalid signature.

$$Boolean\ V \leftarrow \mathbb{V}(Tg, Msg, \sigma_{Msg}) \tag{4.3}$$

4. **Tracebility ($\mathbb{T}$):** It is a function that takes the parameter, tag field $Tg = (issue, pk_N)$, and pair of message and its corresponding signature, (Msg, σ_{Msg}), (Msg', σ'_{Msg}), and generates the output one of the following strings "indep," "linked," or pk_{N_i}, where $pk_{N_i} \in \mathbb{S}|_{pk}$. This can be represented as shown in (4.4).

$$T(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg}) = \begin{cases} indep & if\ N_i \neq N'_i, N_i \in N \\ linked & else\ if\ Msg = Msg' \\ Pubk_{N_i} & otherwise \end{cases} \tag{4.4}$$

For simplicity, we often write $(pk_N, sk_N) \leftarrow Gen(1^k)$ to denote the experiment of $(pk_i, sk_i) \leftarrow Gen(1^k)$ for $i \in N$ and assigning $(pk_N, sk_N) = (pk_i, sk_i)_{i \in N}$. As an ordinary signature scheme, a *TRS* scheme must satisfy the following correctness conditions: For every $k \in N$, every $n \in N$, every $i \in N = 1, ..., n$, every issue $\in 0, 1^*$, and every $m \in 0, 1^*$, if $(pk_N, sk_N) \leftarrow Gen(1^k)$, and $\sigma \leftarrow \mathbb{S}_{sk_i}(L, m)$, where $L = (issue, pk_N)$, it holds with an overwhelming probability (ink) that $\mathbb{V}(L, m, \sigma) = 1$.

E. Why We Need TRS?

TRS is a crucial cryptographic tool used in *CrossLedger* to enhance the security and privacy of cross-chain asset transfers. We have used the *TRS* in *CrossLedger* due to its unique benefits compared to other signature schemes, particularly in the context of decentralized and cross-chain environments. Following are the key benefits of *TRS* over the other signature schemes, which are very important in the context of the cross-chain environment, such as *CrossLedger*.

- **Anonymity with Accountability:** Traditional ring signatures provide a high level of anonymity, allowing a signer to remain indistinguishable within a group of users. However, this anonymity can lead to challenges in accountability, making it difficult to trace malicious activities or prevent double-spending. *TRS*, on the other hand, maintains the anonymity feature but introduces a traceability aspect. This means that while the signer remains anonymous within the ring, their actions can be traced if necessary, enabling the system to link multiple signatures from the same user without revealing their identity. This is particularly important in cross-chain transactions, where accountability must be balanced with privacy.
- **Ensuring the Privacy of Asset Forwarder:** During the trust establishment phase between AFs of different blockchains, *TRS* is used to sign the Asset Forwarder Verification (AFV) message. This message is broadcast across the network of the opposite blockchain to establish trust. The *TRS* ensures that the identity of the asset forwarder remains anonymous within a set of possible signers while still allowing the network to link and verify multiple AFV messages if needed. This enhances the trustworthiness of the asset forwarders without compromising their privacy.
- **Enhanced Security Against Sybil Attacks:** Sybil attacks, where a single entity generates multiple identities to manipulate the network, are a concern in decentralized systems. *TRS* adds a layer of security against such attacks by ensuring that even if a malicious user creates multiple identities, their actions can be traced and linked if they attempt to sign multiple messages, thereby preserving the integrity of the cross-chain transfer process.
- **Ensuring Non-Repudiation and Unlinkability:** *TRS* contributes to non-repudiation by ensuring that once a transaction is signed, the signer cannot deny their involvement, as their signature can be traced if necessary. At the same time, *TRS* supports unlinkability by making it difficult for an adversary to link multiple transactions from the same user unless they attempt malicious activity, such as double-spending. This balance of traceability and unlinkability is crucial for the secure and private transfer of assets between different blockchains.

F. Attack Model

Asset transfer is a very crucial cross-chain communication between two blockchains. A malicious user, say $\mathbb{U}_M$, is always possible to intervene in the communication or present itself as the legitimate user. Here, we consider some assumptions for the security and performance of the system.

- The malicious user $\mathbb{U}_M$ may attempt to exploit the open nature of the communication channel between blockchains. The malicious user might try to compute secret values if a node's private credentials are compromised. However, *CrossLedger* mitigates this threat through the use of secure cryptographic techniques, including *ECC* and *TRS*, which ensure that even if a communication channel is open, the transmitted data remains secure and confidential.

- We have also considered that, the AF could act as malicious asset forwarder denoted as AF_M. The AF_M might attempt to double-spend assets, delay transactions, or intentionally fail to transfer assets to the destination blockchain. However, the *CrossLedger* mitigates this risk through its Asset Forwarder Verification message ($Mesg_{AFV}$), where each AF must obtain verification from a majority of nodes in the opposite blockchain. The *TRS* mechanism ensures that AFs cannot easily forge verification messages, and any discrepancies in behavior are quickly detected and addressed by the network.

- $\mathbb{U}_M$ could attempt to execute a double-spending attack by either spending the same asset twice (discussed in Lemma 6 Case 1) or by creating duplicate/conflicting assets (discussed in Lemma 6 Case 2), i.e., $Owner(\mathbb{U}_M), Asset_Value_A$ is true, where $Asset_Value_A$ belongs to some legitimate user say U_A. *CrossLedger* defends against this by requiring mutual verification and trust establishment between AFs on both blockchains. The *TRS* ensures that only legitimate transactions are processed, making double-spending virtually impossible.

- The $\mathbb{U}_M$ might generate multiple fake identities to manipulate the asset transfer process (discussed in Lemma 8 as Case 1) or attempt to install a fake node as an AF (discussed in Lemma 8 as Case 2). *CrossLedger* counters this by selecting AFs based on their previous performance and verification by other nodes in the network. The use of the *Algorithm* 7 and *Algorithm* 8 ensures that only legitimate nodes can function as AFs, while *TRS* helps verify the authenticity of the AFs.

- The $\mathbb{U}_M$ might attempt to delay the $Mesg_{AFV}$, disrupting the asset transfer process. *CrossLedger*'s security model includes a threshold time parameter, which ensures that any undue delay in $Mesg_{AFV}$ verification results in the transaction being rolled back, thus protecting the system from liveness attacks, i.e., it is not possible due to the atomicity characteristic of *CrossLedger*.

- The attacker model also considered the possibility of collusion among multiple AFs or nodes to manipulate asset transfers. *CrossLedger* addresses this by requiring cross-chain verification of each transaction, where the AFs on each blockchain must independently verify and agree on the transaction's validity. The use of *TRS* further reduces the risk of collusion by linking and tracing transaction signatures, making it difficult for colluding nodes to escape detection.

- **Trust Assumptions:** 1. We have assumed that the majority of nodes in the network act honestly and that cryptographic techniques (such as *ECC* and *TRS*) used are secure. We also assume that initially selected AFs are trustworthy, but this trust is continuously verified through the network's decentralized processes. Any deviation from expected behavior by an AF leads to immediate corrective action, including the possibility of excluding the AF from future transactions.

Additionally, the trust between AFs is established via the *Algorithm* 8, which requires verification from multiple nodes across blockchains. The *TRS* ensures that this process is both secure and traceable, thereby reinforcing the trust assumptions built into the system.

4.2.3 Methodology

The methodology of *CrossLedger* consists of three steps for asset transfer between two different blockchains. The first step is asset forwarder selection, the second step is trust establishment, and the third step is asset transfer and confirmation. In asset forwarder selection, one of the nodes from blockchain networks is selected as an asset forwarder AF in both the blockchains (i.e., blockchain A and blockchain B). After the successful selection of an AF, the next step is establishing trust between these two asset forwarders, i.e., AF_A and AF_B from blockchain A and blockchain B, respectively. To establish trust, the whole network participates in the trust establishment process. When the trust is established, the AF transfers the asset from blockchain A to blockchain B. To describe the proposed scheme, we have considered blockchain A as the source blockchain and blockchain B as the destination blockchain. A detailed description of all three steps is given below.

Step-1: Asset Forwarder Selection (AFS)

The *Algorithm* AFS is straightforward yet very effective. In the proposed asset forwarder selection, we have received help from the blockchain's base consensus algorithm. This means that if blockchain A runs the PoW consensus algorithm, the asset forwarder is selected based on that; similarly, if blockchain B runs the PoS consensus algorithm, the asset forwarder is determined based on the PoS consensus. The *Algorithm* 7 works as follows:

1. The algorithm is simple and takes negligible time to select the AF. Consider AF_A as the asset forwarder of source blockchain A and AF_B as the asset forwarder of destination blockchain B.

2. The current previous miner (CPM) is selected as the AF. There are two conditions to select the CPM as AF.
   1. If the current miner (CM) is not equal to the current previous miner, i.e., if the current miner is not again selected as the miner for the next block generation. Then, the CPM is selected as the AF.
   2. If the CPM is also a CM, then the previous miner (PM) of the CPM (denoted as PM_of_CPM) is selected as the AF. This step can be understood as, let there are three nodes N_1, N_2, and N_3, who generate three blocks B_1, B_2, and B_3, respectively. The blocks are arranged as $B_1 \leftarrow B_2 \leftarrow B_3$. B_3 is the latest block generated by node N_3, and B_2 is the parent block of B_3 generated by the node N_2, in this case the node N_2 is the CPM of node N_3. Now let's suppose block B_2 and B_3 both are generated by the same node N_2 and block B_1 is generated by node N_1. Then, in that case, the node N_1 is PM_of_CPM.

3. When the AF is selected, then a token is generated as $TokenA = createToken(ID_M, T, PBH)$ which includes the ID of CPM, Timestamp, and hash of the previous block PBH. After generating, the token is used for the next step, i.e., trust establishment.

Algorithm 7: Asset Forwarder Selection (AFS) Algorithm

Input: CM: Current Miner, CPM: Current Previous Miner, ID_M: Miner ID of CPM, T: Timestamp, PBH: Previous Block Hash

Output: Asset Forwarder Node

Function `AFS`$(CM, CPM, PM_of_CPM, T, ID_M, PBH)$**:**

- **if** $CM \neq CPM$ **then**
  - $AF \leftarrow CPM$;
  - `/* Current Previous Miner is selected as an Asset forwarder. */`
- **else**
  - $AF \leftarrow PM_of_CPM$;
  - `/* Previous Miner of Current Previous Miner is selected as an Asset forwarder. */`
- **if** $AF.isSelected() = TRUE$ **then**
  - $TokenA = createToken(ID_M, T, PBH)$;
  - $Send_to_BlockchainB(TokenA)$;

Step-2: Trust Establishment and Parameter Exchange

After the selection of the Asset forwarder (AF), the next step is trust establishment between the AFS of both the blockchains, i.e., blockchain A and blockchain B, as described in *Algorithm* 8. To establish trust between both the forwarders, each forwarder broadcasts the signed Asset Forwarder Verification (AFV) message in the network of the opposite blockchain. The AF_A sends the signed AFV message to the network of blockchain B, and AF_B sends the signed AFV message to the network of blockchain A. A TRS is used to sign the AFV message. Each node of the opposite blockchain verifies the signed AFV message. If the AFV message is successfully verified, each node sends the verified message to the AF of the opposite blockchain. If AF_A receives n verification from the network of blockchain B within a timeline, and similarly, AF_B receives the n verification from the network of blockchain A within the timeline, then the trust is established. Here, n is the majority number of honest nodes in each blockchain. Figure 4.2 shows the necessary parameter exchange between asset forwarders after establishing trust between them. This parameter exchange is highly important to understand the block structure and consensus algorithm of both blockchains.

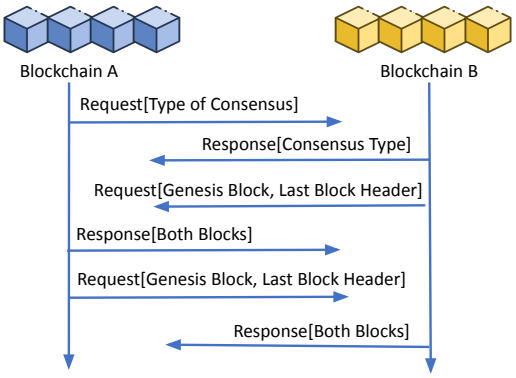


Figure 4.2: Essential parameter exchange between AF_A and AF_B

Algorithm 8: Trust Establishment Algorithm

Input: T: Timestamp, $TokenA, TokenB$: Token of AF_A and AF_B, ID_M: Miner ID of CPM, $send_t$: Sending time of (Msg_{AFV}), $recv_t$: Receiving time of n verification, $thre_t$: Threshold time, $diff_t$: Time difference between $recv_t$ and $send_t$

Output: Trust_Status

Function `TEA(`$T, TokenA, TokenB, ID_M$`)`**:**

```
/* Creation of AFV message                                          */
```
$Msg_{AFV} = Enc_{sk_B}(ID_M||T||TokenA)$;

Blockchain B $\leftarrow send(Msg_{AFV})$;

if $Verification_count(Msg_{AFV}) == n$ ***and*** $diff_t \leq thre_t$ **then**

$Trust_Status.AF = Valid$;

```
/* n is the total number of nodes (sender/receiver)                 */
```

else

$Trust_Status.AF = Invalid$;

Step-3: Asset Transfer and Confirmation (ATC)

This is the third and last step of the scheme. After the successful completion of step-1 (Asset forwarder selection) and step-2 (trust establishment and parameter exchange), the system is ready to transfer an asset from source blockchain A to destination blockchain B. The ATC can only be successfully executed if the sender U_A owns the $Asset_Value$, i.e., An asset has exactly one owner at a time $Owner(U_A, Asset_Value)$ is true if and only if U_A owns $Asset_Value$. Consider U_A and U_B as a domain of users, and asset $Asset_Value_A$, owned by the user U_A, belongs to the set of legitimate assets which can be defined as $Asset_Value_A \in \bigcup_{i=1}^{m} Asset$. Also, consider the $\mathbb{S}|_M$ is the set of miners and any AF is selected if and only if $AF_A/AF_B \in \mathbb{S}|_M$.

We have used Elliptic Curve Cryptography (ECC) and a Secure Hash Algorithm (SHA256) to ensure the confidentiality and authenticity of the asset transfer. The key pair of asset forwarder AF_A is: sk_A, pk_A, and kay pair of asset forwarder AF_B is: sk_B, pk_B. Figure 4.3 shows the sequence of asset transfer communication steps. U_A first sends the asset to its blockchain network. The blockchain network puts the asset in the latest block and also puts that asset in the Asset Transaction Pool (ATP_A). From the ATP_A, the asset is transferred to ATP_B. The AF_B takes the asset from the ATP_B and sends it to the network of blockchain B. Blockchain B first verifies the authenticity of the received asset, and if it is verified, the confirmation is sent to both the users, i.e., sender and receiver. The AF_B also includes the asset transaction in the latest block of blockchain B. This is because, if it is required to track the asset's origin, this block can be used for traceability only with legitimate AF. The node can never trace the block if it is not legitimate AF. Thus, it provides the traceability of the assets. The ATC *Algorithm* 9 works as follows:

1. If the node is AF_A, i.e., asset forwarder of the source blockchain, then it is responsible for the asset transfer. To accomplish this task, AF_A first takes the hash of the asset to be transferred, i.e., $Asset_hash = Hash(Asset_Value)$. After that, it generates the nonce value from the function called $TRNG()$, which is the true random number generator. It also takes the hash of the previous block, i.e., the hash of the previous block of the latest block, which is to be mined. After taking all the values, AF_A create the asset transaction $ATx = (Asset_hash||T||PBH||nonce)$ and send it using ECC encryption and digital signature as $send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$.

Algorithm 9: Asset Transfer and Confirmation (ATC)

Input: PBH: Previous Block Hash, $Hash()$: Hash function using SHA256, ATx: Asset Transaction, B_i: i^{th} Block, T: Timestamp, AF: Asset Forwarder

Output: Asset Transfer

Function `ATC(`ATx, B_i, PM_of_CPM, T, ID_M, PBH`)`**:**

- **if** *node* == *AF(Sender):*AF_A **then**
  - $Asset_hash = Hash(Asset_Value)$;
  - $nonce = TRNG()$; `/* TRNG() is true random number generator */`
  - $PBH = Hash(previous_block)$;
  - $ATx = (Asset_hash||T||PBH||nonce)$;
  - $\sigma_{ATx} = Sign_{sk_A}(ATx, nonce)$;
  - $send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$ to Blockchain B;
- **else**
  - **if** *node* == *AF(Receiver):*AF_B **then**
    - receive$(\sigma_{ATx}, Enc_{pk_B}(ATx, Asset_Value))$;
    - $ATx, Asset_Value = Dec_{sk_B}(ATx, Asset_Value)$;
    - $Asset_hash = Hash(Asset_Value)$;
  - **else if** $Verify(pk_A, ATx, \sigma_{ATx}) == TRUE$ **then**
    - $Asset_hash = Hash(Asset_Value)$;
    - $ATx* = (Asset_hash||T||PBH||nonce)$;
    - **if** $ATx == ATx*$ **then**
      - $Asset_status(valid)$;
    - **else**
      - $Asset_status(not_valid)$;
      - $node.status = bad$;
- **while** $N_i \leq N$ **do**
  - `/* N=total miners, and` $N_i \in \mathbb{S}|_M$ `*/`
  - **if** $count(Asset_status == valid) \geq 75\%$ **then**
    - $blockchain_B.append(CB(Asset_hash, T, PBH, nonce, ATx))$;

2. At the receiving end, the AF_B is responsible for doing all the processing. It first decrypt and verify the received values as $ATx, Asset_Value = Dec_{sk_B}(ATx, Asset_Value)$ and $Verify(pk_A, ATx, \sigma_{ATx})$. If the signature verification is confirmed, then it calculates the asset hash value as $Asset_hash = Hash(Asset_Value)$ and then re-generates the asset transaction as $ATx* = (Asset_hash||T||PBH||nonce)$, after that, it compares both the transactions, i.e., if $ATx == ATx*$, then it is verified that the asset transfer is correct. The AF_B notifies both users about the confirmation of the asset transfer. After the confirmation, the asset is appended into blockchain B as $blockchain_B.append(CB(Asset_hash, T, PBH, nonce, ATx))$. Here, CB is the create block function.

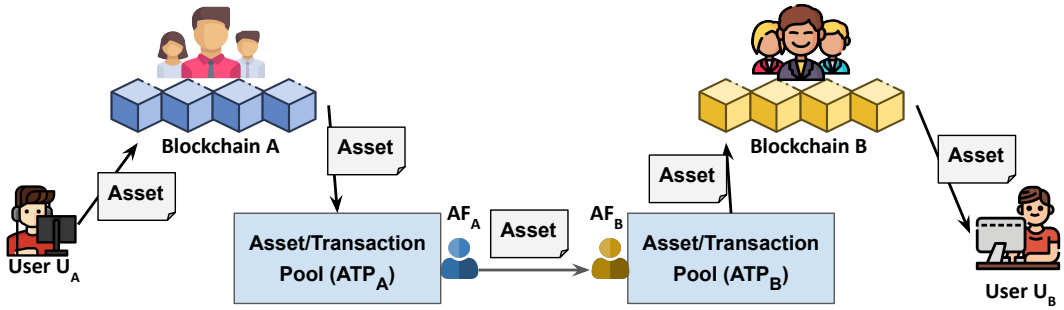


Figure 4.3: Sequence for asset transfer communication from blockchain A to blockchain B

4.2.4 Essential Characteristics of Interoperable Blockchahin

This section explains the characteristics proven by the *CrossLedger*. As mentioned in the previous sections, this scheme supports non-repudiation, unlinkability, confidentiality, atomicity, and interoperability.

Lemma 1 (Non-repudiation)**.** *CrossLedger supports non-repudiation. Legitimate AF are either the sender or the recipient of an asset.*

Proof. *CrossLedger* selects AF_A to transfer the asset from the sender to the receiver blockchain. To transfer the asset, AF_A must go through a few steps in the *Algorithm* 9. The first step is for AF_A to send σ_{ATx} and $Enc_{pk_B}(ATx, Asset_Value)$ to the receiver blockchain's AF_B for asset transfer. The second step is for AF_B to count AF_A's total $asset_status$, which are *valid*. AF_B checks the validity based on signature verification and received parameters. As a result of successful signature verification and a valid $asset_status$ count, AF_A cannot deny asset transfer. If the validity of $asset_status$ is greater than or equal to 75%, then the block is appended to the blockchain, and AF_A gets the acceptance of asset confirmation from the AF_B. So AF_A can not decline because it has not transferred asset to AF_B; similarly, AF_B also can not decline because it has not received asset from AF_A. As shown in Figure 4.1, the sequence number from 1 to 10 shows the complete sequence of asset transfer, where sequence numbers 9 and 10 are the confirmation acknowledgement of asset transfer from AF_B to AF_A and from AF_A to U_A. If the confirmation is not received from the AF_B, the AF_A considers the transfer incomplete due to some error and will be rolled back. Further, after sending the transfer confirmation, AF_B also can not deny the acceptance of the asset. Hence, it is proved that the *CrossLedger* supports non-repudiation. □

Lemma 2 (Unlinkability)**.** *Let there are two distinct assets Ast_{A1} and Ast_{A2} such that $Owner(U_A, Ast_{A1}, Ast_{A2})$. User U_A transfers those assets to U_B and U_C such that $Owner(U_B, Ast_{A1})$ and $Owner(U_C, Ast_{A2})$. Then for any malicious user/adversary U_M, it is impossible to establish a link between both the assets transferred from the same user U_A.*

Proof. In *Algorithm* 9, we have described how the asset is transferred from one blockchain to another, i.e., from source blockchain A to destination blockchain B. As shown in Figure 4.1 there are two users, U_A (sender at blockchain A) and U_B (receiver at destination blockchain B), also $(U_A, U_B) \in \bigcup_{i=1}^{n} U$, meaning that user U_A and U_B both are legitimate users. Let user U_A hold assets Ast_{A1} and Ast_{A2}, which is recorded in the source blockchain A on block B_3. U_A wants to share that asset with U_B and U_C, which is on another

blockchain (blockchain B, our destination blockchain). However, it is not possible for any user to directly transfer the asset because, in *CrossLedger*, only AF is entitled to transfer and receive the asset.

Let some malicious user or adversary U_M want to establish the link between the two distinct assets transferred from the same sender (U_A). It is impossible for U_M for two reasons: first, all the asset transfers are done via AF only, i.e., there is no possibility of direct asset transfer between two users. Therefore, it is impossible to link the two asset transfers from the same sender. Second, it is important to have the address of two distinct assets sent from the same sender to establish the link for U_M. However, in *CrossLedger*, the asset is bundled in the block along with the other information in both the blockchain, i.e., source and destination blockchain. Additionally, as per the *Algorithm* 9, the asset is transferred as follows: First, the hash of the asset is calculated $Asset_hash = Hash(Asset_Value)$. After that, an asset transaction is created as $ATx = (Asset_hash||T||PBH||nonce)$, which is further signed using sk_A as $\sigma_{ATx} = Sign_{sk_A}(ATx, nonce)$ and then it is sent to blockchain B using ECC based encryption as $send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$. It is clear from the above computation that it is impossible to reveal the address of the asset in any phase of the asset transfer. Furthermore, the AF is selected (described in *Algorithm* 7) in every epoch, which further minimizes the possibility of linkability. Thus, it is proved that the *CrossLedger* supports the Unlinkability of two distinct assets from the same sender.

□

Lemma 3 (Confidentiality)**.** *CrossLedger achieves confidentiality while transferring assets between two different blockchains.*

Proof. In Algorithms 7 & 8, we have defined how a AF is selected and how the trust is established between both the AFs of both the blockchains. Also, as mentioned earlier, *Algorithm* 9 described the asset transfer and confirmation process. To transfer an asset while ensuring confidentiality, the AF does the following computation. First, the hash of the asset is calculated ($Asset_hash = Hash(Asset_Value)$). After computing the *nonce* and PBH, the AF create the asset transaction using the earlier computed values (ATx =$(Asset_hash||T||PBH||nonce)$). To ensure the authenticity the AF sign the ATx ($\sigma_{ATx} = Sign_{sk_A}(ATx, nonce)$) and then to ensure the confidentiality, AF encrypt the ATx and $Asset_Value$ using ECC based cryptographic function ($send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$). After that, it sends it to blockchain B. The receiver AF first decrypts the received message and then verifies the signature. If it is verified, it is ensured that the message was not modified during the transfer. Additionally, it is assumed that the private key is not compromised. Thus, the *CrossLedger* supports confidentiality.

□

Lemma 4 (Atomicity)**.** *The CrossLedger is said to have atomicity characteristics if any asset Ast_A belongs to some user U_A is either successfully transferred from one source blockchain A to destination blockchain B or aborted if some error occurs.*

Proof. There are two ways to understand the atomicity in *CrossLedger*.

1. As shown in Figure 4.1 and *Algorithm* 9, when AF_B receives asset transaction from AF_A, it broadcasts this transaction to $Blockchain_B$ network and executes two comparison statements.

(a) The first comparison operation checks the equality between ATx and $ATx*$. If this statement returns true, then the *Asset_status* is marked as valid.

(b) The second comparison operation checks the validity of *Asset_status* is more significant than equal to $(3/4)*n$.

If both comparison statements return true, the transaction is committed in the block and appended to the blockchain. AF_B forward transaction confirmation to AF_A in case of commit otherwise rolled back the transaction. This statement proves that *CrossLedger* is atomic.

2. The atomicity of *CrossLedger* can also be understood as per the ownership property. According to the definition of ownership property, a unique asset belongs to only one owner at a time. Let asset Ast_A belongs to user U_A such that $Owner(U_A, Ast_A)$. The ownership of the asset can be changed if and only if the asset is successfully transferred from source blockchain A to destination blockchain B, for which we followed the *Algorithm* 9. When the asset is successfully transferred, then the asset belongs to only U_B ($Owner(U_B, Ast_A)$). Thus, *CrossLedger* ensures the Atomicity in asset transfer.

□

Lemma 5 (Interoperability)**.** *The cross-chain asset transfer protocol CrossLedger is called interoperable across different types of blockchains if and only if it allows the transfer of assets efficiently between two heterogeneous blockchains.*

Proof. Interoperability is an important challenge when we deal with cross-chain value transfer. *CrossLedger* provides interoperability for the cross-chain asset transfer. Unlike the state-of-the-art schemes where one of the blockchains is fixed (either source or destination), in *CrossLedger*, both the blockchains are not fixed and are of heterogeneous type. Usually, the federation approach is the most preferable approach to enable interoperability. In this approach, both the blockchains (source and destination) rely on the trusted third party to confirm the asset value transfer of one chain to another [144]. However, in *CrossLedger*, the asset transfer in two different blockchains is done following the three novel algorithms (*Algorithm* 7, 8, and 9). In *Algorithm* 7, first, we select the AF out of the existing blockchain network nodes in both the blockchains. After that, using *Algorithm* 8, a trust is established between both the asset forwarders, i.e., between AF_A and AF_B. Therefore, instead of relying on a third party, the *CrossLedger* establishes trust between two selected AFs. Thus, the *CrossLedger* eliminates the dependency on a third party to provide interoperability, which also assures higher security and privacy, as no third party is involved in the communication. The parameter exchange shown in Figure 4.2 strengthens the interoperability. According to Figure 4.2, both the blockchains share the type of consensus, the header of the genesis block, and the header of the last block. It is important to understand the information shared for the asset transfer. Thus, *CrossLedger* provides interoperability. □

4.3 SECURITY ANALYSIS

In this section, we provide a security analysis of our scheme and discuss how *CrossLedger* mitigates or eliminates some well-known attacks against the blockchain.

Lemma 6. *[Double-Spending Attack] The double-spending attack occurs in the cross-chain asset transfer when a* ***Case 1:*** *User U_A wants to spend the same asset twice to two different users, U_B and U_C.* ***Case 2:*** *Malicious user $\mathbb{U}_M$, wants to send a duplicate or conflicting asset,*

Proof. **[Solution] Case 1:** *CrossLedger* is secure against a double-spending attack because trust has been built between the asset forwarders of both blockchains, i.e., blockchain A and blockchain B, which require n verification from opposite blockchains. Here, n is the number of nodes in each blockchain. To get the n verification, the asset forwarder (AF) from each blockchain broadcasts the Asset Forwarder Verification (AFV) message to the opposite blockchain's network to create trust. To validate the legitimacy of the asset forwarder and build trust, a *TRS* is employed to sign the AFV message. As shown in *Algorithm* 8, $if(Verification_count(Msg_{AFV}) = n)$ then $Trust_Status.AF = Valid$. If the AF of blockchain A obtains n verification from the network of blockchain B, and the AF of blockchain B receives n verification from the network of blockchain A, trust is established. Since the trust is established between both forwarders after receiving n verification, the value of Pz is negligible. Therefore, it is clear that the *CrossLedger* is secure against the double-spending attack.

[Solution] Case 2: As we have described above, the definition of ownership of the asset is the leading cause of the prevention of the double-spending attack. Let the user $U_A \in \bigcup_{i=1}^{n} U$ and the asset which the user U_A owns is $Asset_Value_A \in \bigcup_{i=1}^{m} Asset$, for any malicious user, say $\mathbb{U}_M$, such that $\mathbb{U}_M \notin \bigcup_{i=1}^{n} U$ tries to spend asset $Asset_Value_A$ which does not belong to it, then it is not possible because An asset has exactly one owner at a time $Owner(U_A, Asset_Value_A)$ is true if and only if U_A owns $Asset_Value_A$.

□

Lemma 7. *[Liveness Attack] Using liveness attacks, Msg_{AFV} can be delayed as much as possible during the broadcast in the network of the opposite blockchain.*

Proof. The *CrossLedger* is secured from the liveness attack. Let's consider AF from blockchain A, which sends the AFV message to blockchain B's network, and AF of blockchain B, which sends the AFV message to blockchain A's network. To validate the legitimacy of the asset forwarder and build confidence, the AFV message is signed using a *TRS*. If AF of blockchain A gets n verification from the network of blockchain B in the threshold time frame, and AF of blockchain B receives n verification from the network of blockchain A within the timeline, trust is established. If $Verification_count(Msg_{AFV}) == n \&\& diff_t \leq thre_t$ from *Algorithm* 8 satisfied then the $Trust_Status$ of AF is declared as valid. The liveness attack can not affect AF because if the value of $diff_t$ is greater than $thre_t$, there is no trust established between both the AFs. The AF assumes that there has been a liveness attack between two blockchains if the value of $diff_t$ is higher than $thre_t$. Now, AF of one blockchain again forwards the AFV to other blockchains to establish the trust. □

Lemma 8. *[Sybil Attack] The Sybil attack is a major concern in almost every blockchain network. In cross-chain also, the Sybil attack is a big issue to be addressed. There are two cross-chain cases where this attack can take place.* ***Case 1:*** *When a malicious user creates multiple fake accounts and tries to transfer the assets which do not belong to them.* ***Case 2:*** *When a malicious user creates a fake node as asset forwarder AF. We have addressed both cases in the CrossLedger.*

Proof. **[Solution] Case 1:** We have given the ownership definition of any asset to its user. The *Algorithm* 9 also works according to the ownership of the legitimacy of the asset. The ownership of an asset de-

fines who (user) owns the asset. Let the user $(U_A, U_B) \in \bigcup_{i=1}^{n} U$ and the asset which the user U_A owns is $Asset_Value_A \in \bigcup_{i=1}^{m} Asset$. If a malicious user $\mathbb{U}_M$ wants to transfer some asset say $Asset_Value_A$ which belongs to the user U_A, is possible if and only if $Owner(\mathbb{U}_{\mathbb{M}}, Asset_Value_A)$ is true, where $\mathbb{U}_{\mathbb{M}} \notin \bigcup_{i=1}^{n} U$. However, this is not possible because only one user owns any asset at a time. For example, before the transfer of the asset, $Asset_Value_A$ belongs to the user U_A, and after the successful transfer and confirmation, the $Asset_Value_A$ belongs to the user U_B. However, it is mandatory that both the users, U_A and U_B, should be part of legitimate users of the blockchain network, i.e., $(U_A, U_B) \in \bigcup_{i=1}^{n} U$.

[Solution] Case 2: Another use case for the Sybil attack is that if any malicious node tries to become a AF in place of the legitimate node. To address this issue, the two algorithms of *CrossLedger*, *Algorithm* 7 and *Algorithm* 8, are used.

In the first step, the *Algorithm* 7 selects the AF as described in the algorithm. It is to be noted that the selected AF is the one node that has previously successfully added the legitimate block to the blockchain. This can be understood as follows. Let there are three nodes N_1, N_2, and N_3, who generate three blocks B_1, B_2, and B_3 respectively. The blocks are arranged as $B_1 \leftarrow B_2 \leftarrow B_3$. B_3 is the latest block generated by node N_3, and B_2 is the parent block of B_3 generated by the node N_2. Here, the N_2 is the CPM and N_3 is the CM. In this case, the N_2 is selected as the AF. We have selected the CPM as the AF due for two reasons. First, it reduces the load from the CM to do the task of asset transfer. Second, when a block is added successfully by the CPM, and no malicious activity is found, then it ensures the legitimacy of the N_2. Now let's suppose some malicious node, say $\mathbb{N}_M$ poses itself as the AF. However, before initiating the asset transfer, the AF_B of the destination blockchain first establishes the trust with AF of the source blockchain A, where $\mathbb{N}_M$ claims to the AF of it. To establish the trust, the AF_B broadcasts the Asset Forwarder Verification (AFV) message in blockchain A. A TRS is used to sign the AFV message to verify the authenticity of the asset forwarder and establish the trust. If the asset forwarder of blockchain B receives the $3/4^{th}$ verification from the network of blockchain A, then the trust is established. Due to this process, the malicious node $\mathbb{N}_M$, which poses itself as the AF, is identified and discarded by the AF of the destination blockchain B. This way, the *CrossLedger* is secured from the Sybil attack. □

Concluding Remark

This Chapter introduced *CrossLedger*, a novel protocol designed to facilitate seamless interoperability between distinct blockchain networks while ensuring robust security. This solution addresses the critical challenge of enabling cross-chain asset transfers and communication, which is essential for the future of decentralized applications operating across multiple blockchain ecosystems. The chapter demonstrated how *CrossLedger* effectively bridges the gap between different blockchains, laying the groundwork for a more interconnected and versatile blockchain environment. In the subsequent chapter, we shift our focus to the practical application of blockchain technology in smart cities, exploring how blockchain can be leveraged to enhance IoT, ITS, and smart grid systems through tailored solutions.

5

Blockchain Applications

The digital transformation of cities into intelligent hubs of connectivity, often termed 'Smart Cities,' is no longer a vision of the distant future but a rapidly unfolding reality. Interconnected systems like the Internet of Things (IoT), Intelligent Transportation Systems (ITS), and Smart Grids are at the heart of this transformation. These systems, while revolutionary, are not devoid of vulnerabilities. As cities become smarter, the security of their digital infrastructure becomes paramount. Blockchain technology is a decentralized ledger system that promises to strengthen the security framework of these smart city applications, which are given below.

- **Internet of Things (IoT)** encompasses a vast array of devices, from surveillance cameras to smart thermostats, all interconnected and communicating in real-time. While this interconnectivity offers unparalleled convenience and efficiency, it also opens up a Pandora's box of security challenges. Unauthorized access, data breaches, and device tampering are just a few of the myriad threats that loom large over IoT ecosystems. Blockchain, with its immutable records and consensus-driven validation processes, offers a robust shield against many of these threats, ensuring data integrity and thwarting unauthorized access.
- **Intelligent Transportation Systems (ITS)** are revolutionizing urban mobility, optimizing traffic flow, reducing congestion, and enhancing the overall transportation experience. However, the data-driven nature of ITS, which relies on continuous data exchange between vehicles, traffic lights, sensors, and more, is a potential goldmine for hackers. Blockchain can play a pivotal role here. By decentralizing and encrypting transportation data, blockchain can prevent malicious attacks, tampering, and unauthorized data access, ensuring a seamless and secure transportation network.
- **Smart Grids** represent the next frontier in energy management, allowing for dynamic energy distribution based on real-time demand and supply. Yet, they are susceptible to energy theft, unauthorized grid access, and data falsification. Blockchain's decentralized nature can transform the energy sector by ensuring transparent, tamper-proof energy distribution and consumption records. Moreover, blockchain can facilitate peer-to-peer energy trading, allowing consumers to sell excess energy back to the grid or other consumers, all in a secure environment.

5.1 SOLUTION FOR IOT APPLICATION

Using blockchain in IoT applications significantly enhances security by providing a tamper-resistant ledger that records all transactions, including sensitive messages, making it difficult for unauthorized entities to alter data. Blockchain also enables secure authentication and authorization processes, which further reduce the risks of device impersonation and data breaches. Additionally, by integrating blockchain into IoT applications, these systems are transformed from centralized to decentralized, eliminating the risk of a single point of failure—a particularly important benefit in IoT environments where devices are widely distributed. Furthermore, the immutability feature of blockchain ensures that information recorded from IoT devices is accurate and trustworthy, which is critical for applications such as smart cities and healthcare. The consensus algorithms commonly used in blockchain, such as PoW, PoS, and PBFT, are energy-intensive and introduce latency, resulting in low throughput. This makes them unsuitable for real-time IoT applications that require fast data processing and immediate responses. The energy and resource-intensive nature of these algorithms can be particularly challenging for the low-powered devices typically found in IoT applications. Therefore, we have proposed two security methods for IoT applications. First, BLISS, a blockchain based integrated security system for IoT applications, and second, SCAB-IoTA, a secure communication and authentication for IoT applications. The second method actually enhances the first method. In the first method, we apply smart contracts to create secure clusters of IoT devices so that they can communicate securely in a trusted environment. In the second approach, we introduced the angular distance to measure the distance if the device is movable. Apart from the angular distance, the SCAB-IoTA also uses the hybrid cryptosystem, i.e., ECC and AES, for secure authentication and communication. Before delving into the details of both schemes, we first define why we need blockchain in IoT, our motivation and key contributions, the system model, and the adversary model.

5.1.1 Motivation and Contribution

Security is the main issue that restricts the adoption of IoT in social life. However, many researchers have been working to make the IoT a more reliable and secure technology so that it can be entirely adopted in society to make human life more manageable and convenient. Since researchers and developers develop many schemes and methods, but due to IoT devices' constrained environment (i.e., low computational power, low energy), those techniques are not feasible. Thus, an efficient and secure information transmission scheme is required to exchange meaningful information in IoT applications. Blockchain technology is considered a powerful tool for enabling trusted interactions between various components in a decentralized way. Despite this, some challenges emerge during the integration of blockchain technology with IoT, such as high computation, storage, and communication costs. Therefore, focusing on secure communication, efficient authentication, and performance metrics, we proposed an efficient security scheme for secure communication and authentication that is based on a blockchain scheme called BLISS and SCAB-IoTA. The major contributions are as follows:

- The first methodology we have proposed is BLISS, a resource-friendly integrated security solution for IoT applications using blockchain technology and smart contracts. The smart contracts are to be implemented as a rule for cluster association and secure communication for IoT devices. We have created a testbed using 5 Raspberry Pi 4 modules and a computer system also referred as desktop. The Raspberry Pi 4 is used as an IoT device, and the desktop is a cluster head. Apart from the testbed, we have also deployed the scheme on the Sepolia Ethereum testnet, using Remix IDE for development

and MetaMask for network connectivity. This strategic shift to the Sepolia testnet allows us to better evaluate our system under conditions that closely mirror the Ethereum mainnet. Sepolia provides a realistic environment for testing the latency and overall performance of blockchain transactions, which is crucial for assessing the feasibility and efficiency of the BLISS scheme in a real-world scenario.

- The second methodology we have proposed is SCAB-IoTA, a blockchain-based lightweight authentication and secure communication scheme for IoT. We have designed an algorithm for the generation of a one-time session secret key for secure communication between devices. Furthermore, Unlike BLISS, the secure clusters are created based on the angular distance (AD), i.e., they are to be at a certain distance from the controller (called cluster radius R). Similar to BLISS, we have created a testbed for SCAB-IoTA using multiple Raspberry Pi devices and computer systems to show the feasibility of the proposed scheme.

- The performance of both the proposed schemes is measured based on the four parameters, i.e., computation cost, energy consumption, storage cost, and communication cost. We have also provided a comparative analysis with state-of-the-art schemes to show the performance improvement.

- We also provide a security analysis against various cyberattacks such as identification attacks, Sybil attacks, fake node injection attacks, tampering attacks, message replay attacks, man-in-the-middle attacks, botnet attacks, spoofing attacks, and message integrity attacks. The Random Oracle Model (ROM) is also adopted to show the system's security proof for authentication, integrity, and non-repudiation.

5.1.2 System Model

In the proposed mechanism, there are three types of devices: Blockchain Node (BN), Cluster Head (CH), and Cluster Member (M). The BN manages the record of data, which is shared with members of the cluster. All the CHs have an identical copy of the blockchain. The CH has high processing power and efficiency amongst all the cluster's devices (cluster members). Due to this property, the CH controls every cluster thus, we name it as the controller. The functions of the controller device are the authentication of candidates, block creation, and maintaining the blockchain. The controller plays an essential role in the candidate authentication process. The initiation of communication between two candidates is only possible when they belong to the same secure cluster. The IoT devices are smart devices, such as smart bulbs, smart speakers, specialized medical equipment, air-conditioners, and many others. Every cluster is created with a verification process so that no malicious devices can be a part of the cluster, and all the cluster members, referred to as candidates, can communicate with each other in a secure environment. The candidate first needs to send the join request to the controller. After authenticating the candidate, the controller sent some secret credentials to the candidate. So, only candidates having these credentials are called authentic devices. This exchange of information is called a request-response communication. All types of communication are secured with ECDSA (192-bit) and SHA256, which makes forging communication challenging. Every piece of information exchanged must be written into blocks to become a permanent record, including the candidate's authentication information. In the subsequent sections, we have briefly described all the phases. The used mathematical symbols are listed in Table 5.1. Figure 5.1 shows our scheme's framework and the interaction between components. There are three controller devices with ID Ctr_{11}, Ctr_{22}, and Ctr_{33} with cluster IDs Cst_1, Cst_2, and Cst_3, respectively. Ctr_{11} is connected with 4 candidate devices. The request to join the cluster or communicate with other candidate devices is shown in the dashed red arrow color, and

the blue arrow shows the response from the controller to the respective candidate device. We have used the ECDSA for public-key cryptography, and a 192-bit AES is used with secret-key S_K. The BN generates the key-pair (BK_{pub}, BK_{pri}) at timestamp T_1, and the controller generates the key-pair (Ctr_{pub}, Ctr_{pri}) at timestamp T_2. The public key of the BN (BK_{pub}) is shared with the controller because only the controller has the right to create the block and store it on the blockchain node. The public key of the controller (Ctr_{pub}) is shared with the candidate device. The secret key is the one-time session key, which is created and assigned by the controller to the candidate device (C_i).

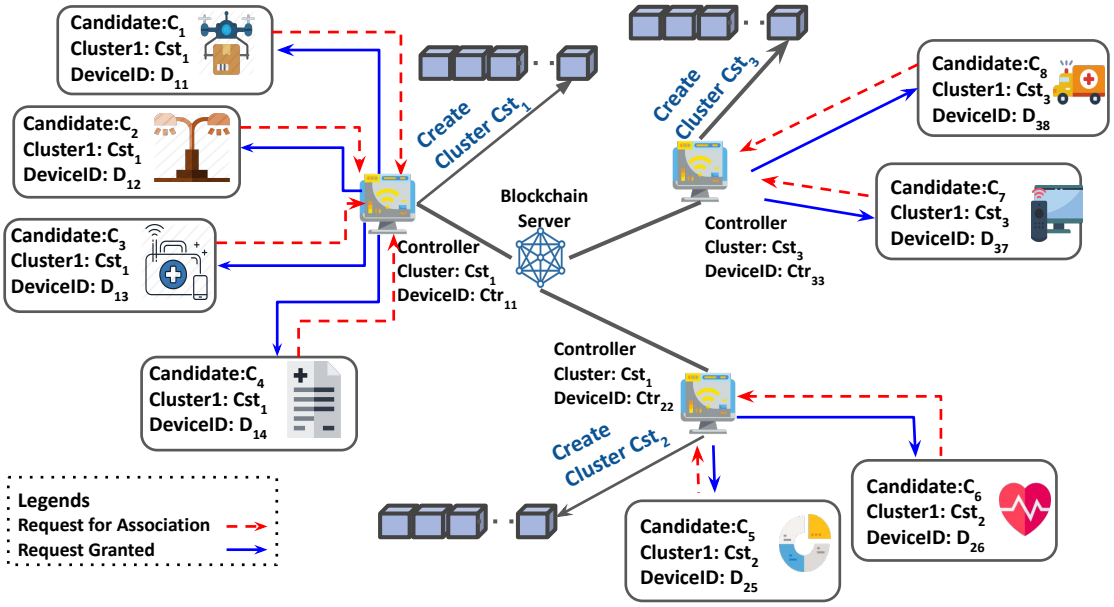


Figure 5.1: System model for IoT application

Table 5.1: Symbol table for BLISS and SCAB-IoTA

Symbol	Description	Symbol	Description
σ_x	Digital Signature of x	$H(x)$	Hash (SHA256) function for the input x
C_i	i^{th} candidate device	D_{ji}	ID of i^{th} candidate device in j^{th} cluster
C^i_{addr}	Unique address (HEX code) of i^{th} candidate device	Cst_i	i^{th} cluster device
Ctr_{id}	Controller's ID	$\triangle T$	Threshold time
C^i_{pub}, C^i_{pri}	Public-private Key-pair of candidate device $\forall$ i = 1,2,...	Ctr_{pub}, Ctr_{pri}	Public-private Key-pair of controller device
BK_{pub}, BK_{pri}	Public-private Key-pair of Blockchain Node	$Req[x]$	Request message with parameter x
$Res[y]$	Response message with parameter y	$M_1, M_2, ..$	Messages
$Tr_1, Tr_2, ...$	Transactions	$Mrt()$	A function representation to create Merkle root

5.1.3 Adversary Model

Adversary modeling is used to identify potential threats or weaknesses in a system and build countermeasures. With adversary modeling, defenders may see what defences are needed, the system's nature, the attacker's profile, and their intended assets. An adversary model is a set of assumptions and proof methods used to assess a test protocol's security. Our adversary model is based on the well-known Dolev-Yao (DY) adversary model [145]. The DY model can express all conceivable attacker actions in a compact manner. This aspect of the DY model makes it a widely used adversary model. In other words, the DY model con-

siders end-users as secure and trusted entities. In the proposed schemes, the information is shared between the candidate and controller over a public channel. Thus, we make some assumptions for the security and performance of the system according to the DY adversary model mentioned above. The properties of the adversary are given below.

1. The adversary $\mathbb{A}$ knows the communication channel is open, and it can compute the secret values if $\mathbb{A}$ has private credentials.
2. Consider an authorized controller performing some malicious activities, e.g., trying to put a fake transaction in the block or a fake block. However, smart contracts take care of this.
3. A legitimate candidate can act as an adversary $\mathbb{A}$ to make a DoS attack by sending multiple fake transactions. However, any transaction is included in the processing after the verification by smart contracts.
4. An adversary can compromise the confidentiality and integrity of the shared transactions in the communication channel. However, all communication between devices is secured using a digital signature (ECDSA) and the hash algorithm (SHA256). The private key is assumed to be unknown to the adversary; hence, an adversary cannot re-sign the transactions.
5. An adversary can not break the cryptographic algorithms (ECDSA) digital signature and one-way hash) and is considered secure.

5.1.4 BLISS: Blockchain-based Integrated Security System for Internet of Things (IoT) Applications

The BLISS is a blockchain-enabled secure storage and communication method to provide integrity to the information and authentication to the participating devices of IoT. BLISS has two phases: first, authentication and cluster creation, and second, candidate-to-candidate communication. A controller must authenticate all the candidates who want to join the cluster and want to interact with each other. All the controllers synchronize to maintain the consistency of the blockchain. Candidates need to be authenticated by a controller to join the cluster. The controller has the following functions: authentication of devices, generating genesis blocks, and mining new blocks. Candidate devices who wish to join the cluster and are waiting to be authenticated with the controller send an authentication request to the controller. After authenticating the device, the controller sends its public key and a signed token to the candidate. So, only those devices that have a signed token are called authentic candidates. Every exchanged information must be written into blocks to become a permanent record, including the candidate's authentication information. Each block contains records or critical information and the hash code of the previous block. The avalanche effect of a hash function states that a minimal change to the input could produce a significant change to the output. Therefore, tampering with information in any block is challenging or nearly impossible.

Authentication and Cluster Creation

The authentication of candidate devices and the creation of a secure cluster of candidate devices are discussed in this section. For the authentication of the device, the controller generates a signed token signed by its private key for every requesting device. As the controller's private key is unknown to anyone, only the controller can generate the token. Hence, no adversary can generate the token, which reduces the chances

of a fake controller device. The requesting candidate device receives the response from the controller after sending the cluster joining request. The candidate device retrieves the confidential information and keeps it for future communication. Figure 5.2 shows the authentication and cluster creation process. In the diagram, the red lines show the requesting line, and the blue line shows the response with the signed token. The process of authentication and cluster creation is described below. Before creating the request to join the

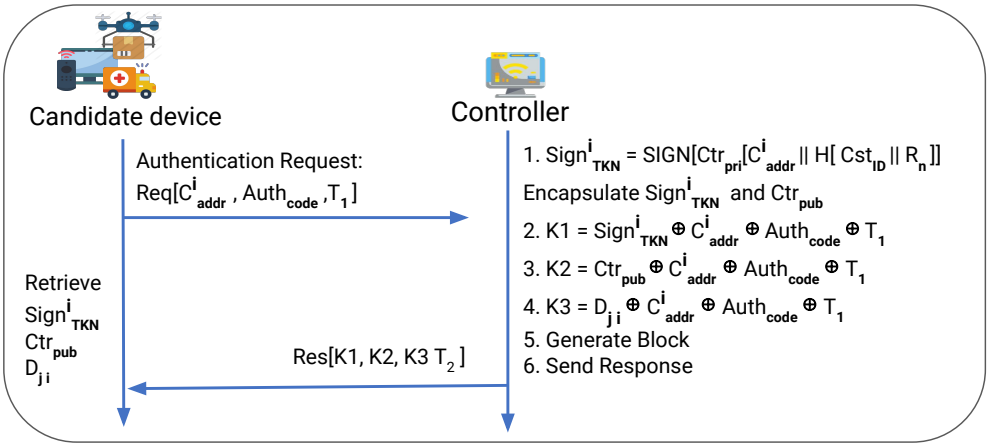


Figure 5.2: Device authentication process in BLISS

cluster, candidate C_i generates its public-private keypair (C^i_{pub}, C^i_{pri}) using the ECDSA as following:

Let G be the generator polynomial on the ECC curve, and select a finite integer number F_n.

1. First it select a pseudo-random integer int in the interval $[1, F_n - 1]$
2. Compute, $Q = int \times G$.
3. C_i's public key C^i_{pub} is Q; and C_i's private key C^i_{pri} is int.

After generating the key pair, it creates its unique address as

$$C^i_{addr} = H(C^i_{pub} \oplus F_n) \tag{5.1}$$

The unique address uniquely identifies every device, whether a candidate or a controller device. After generating the key pair and unique address, it calculates

$$Auth^i_{code} = H(C^i_{addr} \oplus n) \tag{5.2}$$

where $Auth^i_{code}$ is the authentication code, and n is the nonce value. Now, it's time to create the join request. To do this, C_i creates the request as

$$Req[C^i_{addr} || Auth^i_{code} || T_1] \tag{5.3}$$

The request contains the candidate address, $Auth^i_{code}$, and the current timestamp T_1. The $Auth^i_{code}$ is used to check the authenticity of the controller's response. It then sends this request to the controller. If it is the first request from any candidate device, the controller generates the cluster ID, Cst_{ID}

1. $R_n = Math.rand()$

2. Cst_{ID}= $H(R_n||Ctr_{pub})$

The cluster ID is the hash of the random number R_n and the public key of the controller, Ctr_{pub}. Using this method, the cluster ID is unique among all the clusters. After creating the cluster ID, it creates the signed token for the requesting candidate as

$$Sign^i_{TKN} = SIGN[Ctr_{pri}[C^i_{addr}||H[Cst_{ID}||T_2]]] \tag{5.4}$$

The signed token is used to verify whether the candidate belongs to the cluster or not because the token of one cluster is not valid for another cluster. Every cluster will generate a different signed token. This is how it is ensured whether the device belongs to the cluster or not. After creating the signed token, it computes two parameters $K1$ and $K2$ as,

1. $K1 = Ctr_{pub} \oplus C^i_{addr} \oplus T_1 \oplus Auth^i_{code}$
2. $K2 = Sign^i_{TKN} \oplus C^i_{addr} \oplus T_1 \oplus Auth^i_{code}$
3. $K3 = D_{ji} \oplus C^i_{addr} \oplus T_1 \oplus Auth^i_{code}$

The $K1$. $K2$, $K3$ are the intermediate variables used to communicate the controller's public key (Ctr_{pub}), signed token ($Sign^i_{TKN}$), and cluster membership ID (D_{ji}) to the requesting candidate device. The controller also generates the block as the authentication block, which can be used by the other controllers when the device leaves one cluster and joins another cluster. If it is the first block of the system, it generates the genesis block, where the hash of the previous block PBH is set to zero. Otherwise, get the hash of the previous block and generate the block as

$$block = createBlock[T_1, PBH, BH, H(C^i_{addr}), H(D_{ji})] \tag{5.5}$$

The block is appended to the blockchain and sent to the other controllers so that every cluster has a consistent copy of the blockchain. The controller sends $< K1, K2, K3, T_2 >$ to the requesting candidate C^i_{addr}. After receiving the response candidate, C_i first check $\triangle T \geq T_2 - T_1$. If true, it retrieves the secret information from the $K1$ and $K2$ as follows and stores them for future communication.

1. $Get[Ctr_{pub}] = K1 \oplus C^i_{addr} \oplus T_1 \oplus Auth^i_{code}$ = $(Ctr_{pub} \oplus C1_{addr} \oplus T_1 \oplus Auth_{code}) \oplus C1_{addr} \oplus T_1 \oplus Auth_{code}$
   = $(Ctr_{pub} \oplus C1_{addr} \oplus T_1) \oplus C1_{addr} \oplus T_1$
   = $(Ctr_{pub} \oplus C1_{addr}) \oplus C1_{addr}$
   = (Ctr_{pub})
2. $Get[Sign^i_{TKN}] = K2 \oplus C^i_{addr} \oplus T_1 \oplus Auth^i_{code}$ = $(Sign^i_{TKN} \oplus C1_{addr} \oplus T_1 \oplus Auth_{code}) \oplus C1_{addr} \oplus T_1 \oplus Auth_{code}$
   = $(Sign^i_{TKN} \oplus C1_{addr} \oplus T_1) \oplus C1_{addr} \oplus T_1$
   = $(Sign^i_{TKN} \oplus C1_{addr}) \oplus C1_{addr}$
   = $(Sign^i_{TKN})$
3. $Get[D_{ji}] = K3 \oplus C^i_{addr} \oplus T_1 \oplus Auth^i_{code}$

Now, candidate C_i and the controller are both connected. Similar process is followed for all the devices, i.e., $i = 1, 2, 3, ...$

To join the cluster, the candidate device must execute the smart contract defined in the *Algorithm* 10. The uniqueness of the candidate and controller device is verified using the smart contract. The joining is aborted if any of the rules are violated in the contract. The *Algorithm* 10 defines the contract for joining the cluster. This contract is executed on the controller's end. First, it checks the identity of the controller receiving the cluster joining request from the candidate device. Second, it checks the identity of the candidate device that sent the joining request, and the third condition is to check the signed token of the candidate, which the candidate has received from the controller before sending the joining request, i.e., the authentication step. The signed token is used to verify whether the candidate belongs to the cluster or not because the token of one cluster is not valid for another cluster. This token is signed using the private key of the cluster head, i.e., the controller. It is verified by the public key of the controller as $b \leftarrow VERIFY(Ctr_{pub}(Sign^i_{TKN}))$. If b is found true, it is considered a valid candidate. This contract self-executes as soon as any authentic candidate device sends the joining request.

Algorithm 10: Smart Contract to Join Cluster

Input: $C^i_{addr}, D_{ji}, Cst_{ID}$
Output: Device Validity
Function `joinCluster`$(C^i_{addr}, D_{ji}, Cst_{ID}, Sign^i_{TKN})$**:**

```
if isAvailable(C^i_addr) != true then
    Discard request;
else
    /* the C^i_addr is true for both, candidate and controller */
    if C_i = controller then
        if isAvailable(Cst_ID) != true then
            Discard request;
    else
        if C_i = candidate then
            if isAvailable(Cst_ID) != true then
                Discard request;
            if Sign^i_TKN.Verify = fail then
                Discard request;
            if isAvailable(D_ji) = fail then
                Discard request;
```

Key Distribution and Secure Communication

The key distribution phase is distributing the candidate device's public key to the other cluster members for communication. However, instead of directly sending a key to other devices, the candidate devices store it at the controller. The controller sends the candidate device's public key, say C_1, to the other candidate device, say C_2. The benefit of distributing public keys using this method is that only authenticated can-

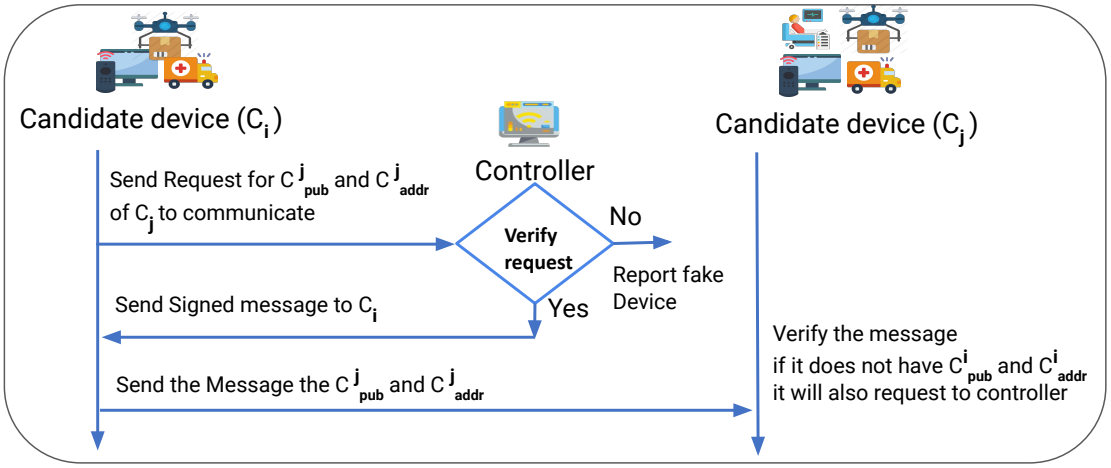


Figure 5.3: Candidate-to-candidate secure communication in BLISS

didate devices can get the key, reducing the possibility of key compromise. The key distribution phase is mandatory for the communication phase because 1) the sender's public key is required to verify the other device's incoming message. 2) to send the secret message to the other candidate device. All the candidate devices have the controller's public key during the authentication and cluster creation phase. The abstract of secure communication between two candidate devices is illustrated in Figure 5.3. The procedure for key distribution and communication is as follows.

Let candidates C_i and C_j want to communicate with each other. First, C_i sends request M_{s1} (a message which contains the request) to the corresponding controller to get public-key C^j_{pub} and address C^j_{addr} of C_j signed with private key C^i_{pri} as $\sigma_{M_{s1}} = SIGN(C^i_{pri}(M_{s1}))$, $Req[C^i_{addr}||D_{ji}||Sign^i_{TKN}||\sigma_{M_{s1}}||T_1]$. Here, $M_{s1} = get(C^j_{pub}$ and C^j_{addr} of $C_j)$. The controller receives the request and first checks whether it is a new request or an old one as $\triangle T \geq T_2 - T_1$, and if it is true, then the controller verifies the request as $b \leftarrow VERIFY(C^i_{pub}, M_{s1}, \sigma_{M_{s1}})$. It also checks the authenticity of C_i against the stored value of C^i_{addr}, D_{ji}, and $Sign^i_{TKN}$. If the signature is valid, the controller checks the identity of both the candidates C_i and C_j. The controller searches for the authenticity of both the candidate devices in the blockchain ledger. If the candidate is authenticated before the communication, then the candidate's public address and ID should already be present on the ledger. If both C^i_{addr} and C^j_{addr} are identified, the controller responds as follows. $M_{s2} = C^j_{pub} \oplus C^i_{addr} \oplus T_2$, $M_{s3} = C^j_{addr} \oplus C^i_{addr} \oplus T_2$, $\sigma_{M_{s2}} = SIGN(Ctr_{pri}(M_{s2}))$, and $\sigma_{M_{s3}} = SIGN(Ctr_{pri}(M_{s3}))$. After signing the response, send it to the requesting candidate, i.e., C_i, as follows. $Res[M_{s2}, M_{s3}, \sigma_{M_{s2}}, \sigma_{M_{s3}}, T_2, T_3]$.

After getting a response from the controller, candidate C_1 first checks whether the response was received in due time or an old response using $\triangle T \geq T_4 - T_3$ where T_3 is the request sending timestamp and T_4 is the received response timestamp. If the response is received within the threshold value, it is further processed; otherwise, it is discarded. After checking the timestamp, the received response's signature is verified, and if it is found to be true, retrieve the C^j_{pub} and C^j_{addr} as $Get(C^j_{pub}) = M_{s2} \oplus C^i_{addr} \oplus T_2$ and $Get(C^j_{addr}) = M_{s3} \oplus C^i_{addr} \oplus T_2$ }.

After retrieving the required values, i.e., C^j_{pub} and C^j_{addr}, C_i send message M_1 after signing it as σ_{M_1} = $SIGN(C^i_{pri}(M_1))$. The C_j receives the signed message, and first, it verifies using C^i_{pub}, and if it is not available, C_j requests from the controller to get it. Then check if $VERIFY(C^i_{pub}(M_1), \sigma_{M_1})$ is valid, then accept it; otherwise, reject it. All the candidates put the messages (treated as transactions) in the transaction pool TP. The controller takes the transactions from the TP as $TP \leftarrow Tr_1, Tr_2, ..., Tr_n$ to create the block on the regular interval. The Merkle root is calculated as the transaction from all the transactions taken from the pool. The Merkel root is added in the block's header, used to check the integrity of the transactions. To propose the block, the controller computes the Merkle root, the hash of the previous block PBH and the hash of the current block BH, and the T_3 as follows:

1. $Mrt = createHashTree(H(Tr_1), H(Tr_2), ..., H(Tr_n))$
2. $PBH = H(LB)$ {LB: last block of blockchain}
3. $BH = H(T_3, PBH, Mrt, H(Tr_1), H(Tr_2)...H(Tr_n))$
4. $block = createBlock[Mrt, BH, PBH | H(Tr_1), H(Tr_2)...H(Tr_n)]$

All the controllers agreed on this new proposed block and added it to the current blockchain. The *Algorithm* 11 illustrates the smart contract for two candidate devices' communication. For successful communication, the smart contract needs to be executed successfully. To summarize, depending on the device type, i.e., the controller or candidate device, the smart contract's rules are applied as follows:

1. **Controller** can build a single trusted cluster with a unique cluster identifier, which should not exist or be active in the blockchain network. The controller's role is to sign the token for the candidates and perform block generation and verification. The controller can not create the two clusters simultaneously, i.e., one controller can only be part of one cluster.
2. **Candidate** is associated with a single cluster only if it exists. Candidates can not belong to two separate clusters at the same time. All the candidates in the cluster require authentication from the controller before any communication. Every candidate must have a signed token signed by the controller.

Token and Key Revocation in BLISS

In the BLISS scheme, maintaining the integrity of both signed tokens and cryptographic keys is paramount. If either becomes compromised, i.e., token or keys, the scheme allows for their immediate revocation and replacement. Here, we present the mathematical description of the token and key revocation process.

Let $\mathbb{CTR}$ denote the set of all controllers, and $\mathbb{C}$ the set of all candidate devices within the BLISS network. Each candidate device $C_i \in \mathbb{C}$ is associated with a signed token $Sign^i_{TKN}$ issued by a controller $Ctr \in \mathbb{CTR}$ and a pair of cryptographic keys (C^i_{pub}, C^i_{pri}), (Ctr_{pub}, Ctr_{pri}) generated by all the legitimate candidate and controller device. The revocation process is illustrated as follows.

1. Upon the compromise of $Sign^i_{TKN}$ or (C^i_{pub}, C^i_{pri}), controller Ctr initiates the revocation process. The controller adds the compromised token or key to a revocation list, R_c for tokens and K_c for keys, both maintained on the blockchain ledger L.

Algorithm 11: Smart Contract for Communication

Input: D_{ji}, Cst_{ID}, $Sign^{i}_{TKN}$, $Sign^{j}_{TKN}$
Output: Device Validity
Function `clusterCommunication`(C^{i}_{addr}, D_{ji}, Cst_{ID}, $Sign^{i}_{TKN}$, $Sign^{j}_{TKN}$):

```
if isAvailable(D_ji(sender)) != true then
    if isAvailable(D_ji(receiver)) != true then
        Discard request;
/* Checking the ID of cluster whether it exists or not                    */
if isAvailable(Cst_ID) != true then
    Discard request;
if sender.Sign^i_TKN.verify = fail then
    if receiver.Sign^j_TKN.verify = fail then
        Discard request;
```

2. Verification post-revocation involves checking any token or key against R_c and K_c in L. A device is considered a valid member of the cluster if both its token and public key pass verification.

Traditional security systems rely on certificate revocation to maintain trust within the PKI framework. Revocation is managed using CRLs (Certificate Revocation Lists) or OCSP (Online Certificate Status Protocol), ensuring that any entity can verify the validity of a certificate. The BLISS scheme's revocation process mirrors these methods by maintaining a revocation list on the blockchain ledger for both signed tokens and cryptographic keys. Just as with CRLs, these lists can be checked by any participant in the network to verify the revocation status of a token or key.

The main differences lie in the medium of distribution and the technology used to implement these revocation lists. While traditional systems may use centralized servers or databases to store and distribute CRLs, the BLISS scheme utilizes blockchain technology's decentralized and immutable nature. This not only ensures that revocation lists are tamper-proof and always up-to-date across all nodes but also leverages the blockchain's inherent security and consensus mechanisms to enforce the revocation status network-wide. Thus, while the principles are similar, the implementation and underlying technologies differ, with BLISS adopting a blockchain-centric approach to revocation in contrast to the centralized models typically used in traditional PKI systems.

5.1.5 SCAB-IoTA: Secure Communication and Authentication for IoT Applications Using Blockchain

SCAB-IoTA is the second blockchain-based security solution for IoT applications. Following the similar architecture of BLISS, SCAB-IoTA enhances the performance and security compared with BLISS as well as the state-of-the-art schemes. Similar to BLISS, a different trust mechanism of cluster creation is employed in SCAB-IoTA. Before joining the cluster, every device must go through a secure authentication mechanism, and only authenticated devices can communicate with others. Unlike BLISS, we have used hybrid cryptography to provide a reliable security mechanism. The public-key cryptography is used for cluster member (D_{ji}) to cluster head (i.e., controller) communication, which is an ECDSA-based encryption/de-

cryption method. Secret-key cryptography is used for member-to-member ($M2M$) communication, which is an AES-based encryption/decryption method. In the cluster creation and member authentication phase, the clusters are created based on the angular distance (AD), i.e., the device that wishes to join the secure cluster first needs to be at a certain distance (called cluster radius R) from the controller. Once the device is in the range of the cluster, it sends a request (which contains device ID C_i) to join the cluster. The controller verifies the device and sends its public key Ctr_{pub} to the device. After getting the D_{ji} and Ctr_{pub}, the device becomes the cluster member. The controller also creates the block at the blockchain node containing the device ID of the requesting device for future authentication purposes. In the secret-key generation and M2M communication phase, the M2M communication is secured with AES encryption using secret-key S_K. Any member of the cluster who wants to communicate with another member must have the secret key generated. At first, The C_i with D_{ji} (i^{th} member of j^{th} cluster) sends a request for communication to C_k (K^{th} member of same cluster). C_k sends a second request to the controller to check for the authenticity of C_i. The controller verifies the member's identification through the blockchain node if the member's block exists, which means the member is authentic. Otherwise, a warning message is generated to the requesting device C_k. After verifying the C_i, the controller generates a one-time secret-key S_K and sends it to both the members, i.e., C_i and C_k. After getting a one-time secret key, members communicate using AES symmetric encryption.

Cluster Creation and Device Authentication

The clusters are created based on the angular distance (AD), i.e., the device that wishes to join the cluster to become a cluster member first needs to be at a certain distance (R) from the controller. Once the device is in the range, it sends a request (which contains device ID C_i) to join the cluster. The controller verifies the device and sends its public key Ctr_{pub} to the device. The device becomes the cluster member after getting Ctr_{pub} and the cluster membership ID (D_{ji}). The controller also creates the block at the blockchain node containing the device ID. *Algorithm* 12 defines how to calculate the distance between two coordinates. It takes two point coordinates as an argument and determines the distance. The device $Point_d$ is having coordinates (X_d, Y_d) and controller $Point_{Ctr}$ is having coordinates (X_{Ctr}, Y_{Ctr}).

Algorithm 12: Algorithm to Find Angular Distance

Input: P_d, P_{Ctr}: are the original coordinates, and $P_d^{'}$ is the coordinate after the movement, concerning their original coordinates.

Output: $d_{i,j}$: It is the angular distance between device and controller.

$P_d \leftarrow X_d, Y_d$; `/* Point coordinates of device */`

$P_{Ctr} \leftarrow X_{Ctr}, Y_{Ctr}$; `/* Point coordinates of controller */`

Function `AD`$(P_d, P_{Ctr}, P_d^{'})$**:**

$$\begin{bmatrix} X_d^{'} \\ Y_d^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_d \\ Y_d \\ 1 \end{bmatrix}; \quad \texttt{/* calculate new coordinates of point } P_d \texttt{ */}$$

$d_{Ctr,d} \leftarrow |X_{Ctr} - X_d^{'}| + |Y_{Ctr} - Y_d^{'}|$;

The Cluster creation and device authentication process is illustrated in Figure 5.4, which is as follows:

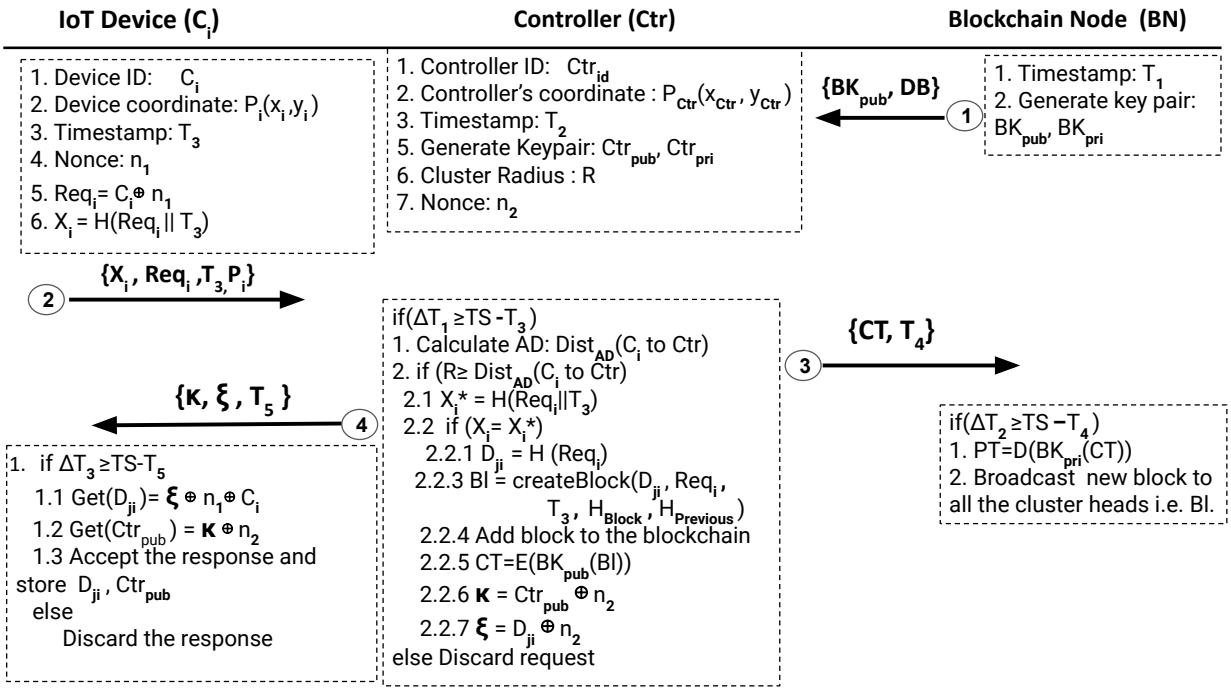


Figure 5.4: Cluster creation and device authentication

1. The device having device ID C_i, device coordinates $P_i(X_i, Y_i)$, timestamp T_3 and a nonce n_1. It create a request to join the cluster as $Req_i = C_i \oplus n_1$, $X_i = H(Req_i||T_3)$ and send $\langle X_i, Req_i, T_3, P_i \rangle$ to the controller.

2. After receiving the request, controller first check if $\triangle T_1 \geq TS - T_3$ is true, otherwise discard the request. If the condition holds true, the controller computes the distance using *Algorithm* 12. If the device is in the range of the controller, then it proceeds further and computes $X_i^* = H(Req_i||T_3)$, then check whether $X_i^* = X_i$. If so, then it does the computation as $D_{ji} = H(Req_i)$, where j is the cluster number. To propose the block, the controller computes the Merkle root, the hash of the previous block PBH and the hash of the current block BH, and the T_3 as follows:

   (a) $PBH = H(LB)$ {LB: last block of blockchain}
   (b) $BH = H(T_3, PBH, D_{ji}, Req_i)$
   (c) $block = createBlock[D_{ji}, Req_i, T_3, PBH, BH]$

   After that, it encrypts the created block using the public key BK_{pub} of the blockchain node as $CT = E(BK_{pub}(block))$ and sends $\langle CT, T_4 \rangle$ to the blockchain node to share with other controllers. Also, the controller computes κ and ξ as $\kappa = Ctr_{pub} \oplus n_2$, $\xi = D_{ji} \oplus n_2$ to share its public key, and device ID with requesting device. Then it sends $\langle \kappa, \xi, T_5 \rangle$ to the requesting device.

3. The blockchain node receives the encrypted new block from the controller. First, it checks whether or not $\triangle T_2 \geq TS - T_4$ is true. If true, then it decrypts the CT using its own private key BK_{pri} as $PT = D(BK_{pri}(CT))$, then it securely broadcasts this new block to all the controllers.

4. After receiving response from the controller, device C_i first calculate if $\triangle T_3 \geq TS - T_5$ is true or not. If true, then it retrieves the public key of the controller from the received response as $Get(D_{ji}) =$

$\xi \oplus n_1 \oplus C_i$, $Get(Ctr_{pub}) = \kappa \oplus n_2$ and store the D_{ji},Ctr_{pub} for future communication. The D_{ji} is now used as the main identity of the C_i for the communication. Other devices follow this procedure, and we create a secure cluster of devices where every device is authenticated from the controller.

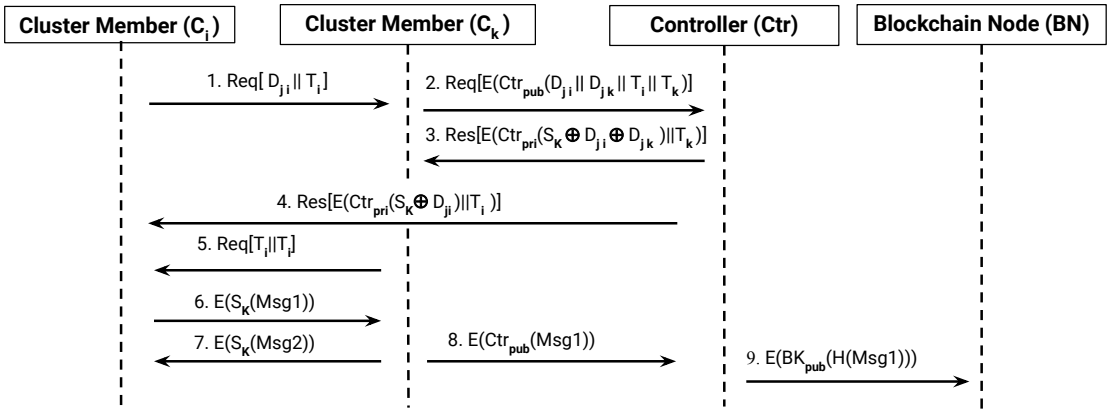


Figure 5.5: Secret-key share and M2M communication

Secret-key Generation and M2M Communication

Communication between two devices is secured using a 192-bit AES symmetric key cryptography. After receiving a request from the member device (D_{ji}) for communication, the controller generates the shared secret key. Communication between member devices and controllers is secured using the ECDSA asymmetric key cryptography. Member D_{ji} who wants to communicate with another member D_{jk} must have the secret key generated by the controller. The secret key generation process is explained in *Algorithm* 13, and the M2M communication is illustrated in Figure 5.5. The process of this phase is as follows:

1. Let, there are two devices C_i and C_k with cluster member IDs D_{ji} and D_{jk} who wants to communicate. Cluster member D_{ji} (i^{th} member in j^{th} cluster) sends a request as $Req[D_{ji}||T_i]$ to D_{jk} (K^{th} member of same cluster) for the communication.

2. Member D_{jk} receives the request and first checks if $\triangle T_1 \geq T_i^* - T_i$ holds; if yes, then, to check the authenticity of the member D_{ji} and one-time secret key generation, it sends a request to the controller as $Req[E(Ctr_{pub}(D_{ji}||D_{jk}||T_i||T_k))]$ and wait for the response.

3. After receiving a request from the D_{jk}, the controller first checks $\triangle T_2 \geq T_j^* - T_j$ is true, then it verifies both the member, i.e., $D_ji \& D_{jk}$ both, from the blockchain database. After verifying both the members first, it generates the one-time secret key S_K using *Algorithm* 13 and sends the response to both the members as $Res[E(Ctr_{pri}(S_K \oplus D_{ji} \oplus D_{jk} \oplus)||T_k)]$, $Res[E(Ctr_{pri}(S_K \oplus D_{ji})||T_i)]$ to D_{jk} and D_{ji} respectively. If the controller does not find either of the requesting members, a warning message is generated so other controllers are notified.

4. Member D_{ji} computes the received response and gets the one-time secret key S_K as $A = D(Ctr_{pub}($ $E(Ctr_{pri}(S_K \oplus D_{ji})||T_i)))$ and then check if $\triangle T_2 \geq T_i^* - T_i$ is true or not, if true then compute $Get(S_K) = A \oplus D_{ji}$ and store it for the communication. Similarly, the member D_{jk} also retrieves the one-time secret key S_K from the received response.

5. The D_{jk} sends a response to D_{ji} as $Res[T_i||T_j]$ as an acknowledgement, i.e., it is ready to accept the next message.

6. Now D_{ji} creates message $Msg1$ and encrypts it using a one-time secret key (using AES encryption technique) as $E(S_K(Msg1))$ and sends it to the D_{jk}.

7. D_{jk} receive the encrypted message and decrypt using the same key S_K, then it creates a message $Msg2$ and sends it to the D_{ji} after encrypt it as $E(S_K(Msg2))$. Similarly, the communication proceeds between two cluster members.

Algorithm 13: Secret-key-Generation

Input: Q_n, R_n
Output: S_k: It is the one-time secret-key
$Q_n \leftarrow$ randomNumber(); $R_n \leftarrow$ randomNumber();
Function `SKG(`Q_n, R_n`)`**:**
  if Q_n *is Prime* **then**
    if R_n *is Prime* **then**
      $N \leftarrow (Q_n - 1) \times (R_n - 1)$;
      $E \leftarrow$ randomNumber() mod 1000;
      $S_k \leftarrow N^E$; `/* The secret key` S_k `is calculated */`
      return S_k;
    else
      Call SECRET-KEY-GENERATION recursively;
  else
    Call SECRET-KEY-GENERATION recursively;

The *Algorithm* 13 is used to generate the secret key. In this algorithm, we first generate two random numbers, Q_n and R_n, and then check whether they are prime numbers. If they are prime numbers, then both numbers are subtracted by one and multiplied by each other. After that, we generated a random number and took mod with 1000 as $E \leftarrow randomNumber()\%1000$, then computed the secret key as $S_k \leftarrow N^E$, which is to be shared with the members. After getting a one-time secret key, both members communicate using symmetric key encryption.

5.1.6 Implementation Detail and Performance Analysis

We have used multiple Raspberry Pi 4 models to implement both schemes. We have also implemented the BLISS scheme on the Sepolia Ethereum testnet, using Remix IDE for development and MetaMask for network connectivity. The performance analysis of both schemes is done on four parameters, i.e., computational cost, energy consumption, storage cost, and communication cost.

Implementation Detail

In BLISS and SCAB-IoTA, we have used the Raspberry Pi 4 (RP4) model with a processor speed of 1.5GHz, memory of 4GB, and an operating system of Raspbian v5.10.17, a Debian-based OS. The Desktop PC has a

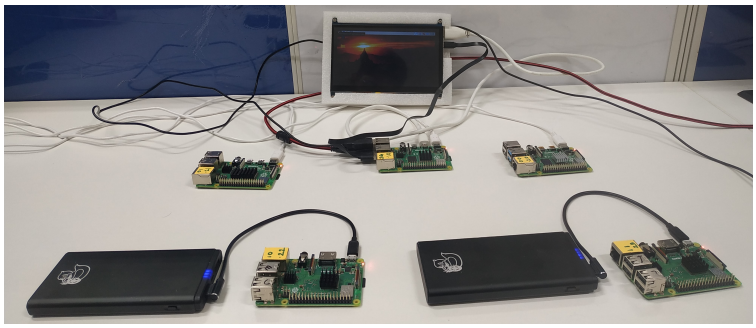


Figure 5.6: Testbed for one cluster of 5 Raspberry Pi model B devices

4.1GHz core i5 processor, 8GB of memory, and Windows 11 as an operating system. We have used Oracle Java 8 and NetBeans 11 IDE for the programming. The testbed consists of 5 Raspberry Pi devices and one computer system. This configuration was deliberately chosen to simulate a realistic and manageable IoT network that allowed us to monitor and accurately measure the system's performance. The testbed is shown in the Figure 5.6. We have used the Solidity programming language in BLISS to implement the smart contract. The functions that we have defined in this smart contract are *createDevicePool*, *createClusterPool*, *isAvailableDeviceID*, and *isAvailableClusterID*. We have also defined two events: *ClusterFound* and *DeviceFound*. These functions and events are necessary to auto-trigger the other two smart contracts. The cost was measured in terms of computational cost, energy consumption, storage cost, and communication cost at the scheme's operation, mainly during the initial authentication process of the devices in each cluster and during the execution of transactions. Power consumption was rigorously recorded throughout the experiment, especially during the cluster formation stage to capture the energy demands of setting up the network and during active communication between the devices to assess the ongoing operational efficiency. Apart from the testbed, we have also deployed the contracts on the Sepolia Ethereum testnet, using Remix IDE for development and MetaMask for network connectivity for the BLISS proposed scheme. This strategic shift to the Sepolia testnet allows us to better evaluate our system under conditions that closely mirror the Ethereum mainnet. Sepolia provides a realistic environment for testing the latency and overall performance of blockchain transactions, which is crucial for assessing the feasibility and efficiency of the BLISS scheme in a real-world scenario. This environment also exposes our contracts to more variable network conditions, including transaction delays and block processing times, which are essential factors in understanding the practical implications of deploying such a system. Using Remix IDE, we have developed and tested our smart contracts effectively, ensuring their integrity and robustness. MetaMask's integration has facilitated a seamless connection to the Sepolia testnet, allowing for efficient contract deployment and interaction. We utilized the Web3.js library to programmatically fetch and analyze block and transaction data from the Sepolia testnet to gather accurate and current statistics. This approach allowed us to meticulously compute the average block creation time and transaction delay over a significant period, ensuring the reliability of our findings. Based on our analysis, we deduced that the average block creation time on the Sepolia testnet is approximately 8-10 seconds. This slightly faster block time compared to the Ethereum mainnet, which averages around 13-15 seconds, is attributable to the lower network congestion typically experienced on the testnet. Additionally, we observed that transaction delays were generally shorter on Sepolia, reinforcing the suitability of this testnet for evaluating the BLISS scheme under less congested network conditions. These insights have been instrumental in enhancing our understanding of how the BLISS scheme performs in a dynamic blockchain environment and have been included in the revised manuscript. Below in Figure 5.7 shows the metamask interface, and Figure 5.8 shows the output of the contract JoinCluster.

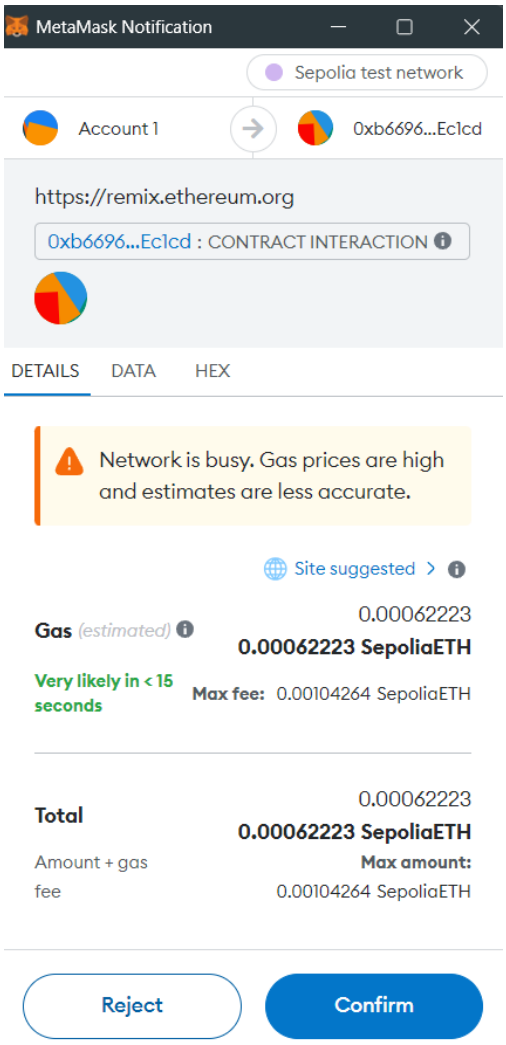


Figure 5.7: Metamask interface to connect with Ethereum testnet

Computational Cost and Energy Consumption

Computational time is the time taken by the proposed schemes for the execution of the whole framework. We have measured this cost in milliseconds (MS). The computational cost is denoted as T_{BLISS_ct} for BLISS and $T_{SCAB-IoTA_ct}$ for SCAB-IoTA. We illustrate the computational cost based on the operational cost for authentication, verification, and communication. Table 5.2 shows the execution time of different cryptographic functions used in the proposed schemes. Table 5.3 show the comparative computation cost of BLISS and SCAB-IoTA with state-of-the-art schemes. It is clear from Table 5.3 that the number of cryptographic operations in BLISS and SCAB-IoTA is lower than the state-of-the-art schemes; therefore, the computation cost is also lower in BLISS and SCAB-IoTA. Despite the lower computation cost in SCAB-IoTA, it is slightly higher compared with the BLISS scheme. The energy consumption is the energy consumed by the system to perform the operations of the scheme. In the execution process, the system requires some energy to carry out all the phases of the scheme. The energy consumption is calculated as $E_{rg} = T_{ct} * C_P$. Here, C_P is a CPU maximum power (10.88 W for a laptop and 2.4W for Raspberry Pi 4), and T_{ct} is a computational time.

```
status: 0x1 Transaction mined and execution succeed
transaction hash:
0x4904b196d7d61cd911be96ba81403a6844643b858158d644cb0697bf968ddd2b
block hash:
0x5e8bf8c4aa1b47e650b76b3c3a62b864aa9fddaf2498744abc9af6fcc5a4aa69
block number: 37 L
from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to: Common. isAvailableDeviceID(uint256)
0xddaAd340b0f1Ef65169Ae5E41A8b10776a75482d
gas: gas
transaction cost: 27927 gas
execution cost: 6495 gas
input: 0xff0 ... cabab L
decoded input:
           {
           "uint256 deviceId":
"687646025091532448527707505288253954804449848235"
           }
decoded output:
           {
           "0": "bool: true"
           }
logs: {
      "from": "0xddaAd340b0f1Ef65169Ae5E41A8b10776a75482d",
      "topic":
"0xa66d51529904c6cff23539df1e611d67015b0840db58d5a936da2dd7c56ccc15",
      "event": "DeviceFound",
      "args": {
            "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
            "owner": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
            }
      }
```

Figure 5.8: Output of the JoinCluster contract. The function call isAvailableDeviceID shows the device is found, mentioned under the decoded output

It is measured in Joules. Earlier, we had already calculated the computational cost. We calculated the energy consumption on two platforms, i.e., the computer system and Raspberry Pi 4 model. Table 5.3 shows the comparative result of energy consumption of BLISS and SCAB-IoTA compared with state-of-the-art schemes. It is very clear from the figure that in BLISS and SCAB-IoTA, energy consumption is reduced compared to state-of-the-art schemes.

Table 5.2: Individual computation time of different cryptographic functions for IoT testbed

Functions	**On Desktop**	**On Raspberry Pi 4GB**
SHA256 Hash function: $T_{H()}$	0.0008	0.005
ECC Crypto-system: T_{ECC}	0.053	1.6
ECDSA signature: T_{sign}	0.071	2.94
ECDSA verification: T_{veri}	0.065	2.86
point multiplication: $T_{mul(.)}$	0.117	5.0
AES Crypto-system 192-bit: T_{AES}	0.125	5.96
Modulo operation: $T_{mod(.)}$	0.067	2.0

Table 5.3: Computation cost (CC) and energy consumption (EC) of BLISS, SCAB-IoTA, compared with state-of-the-art schemes on Desktop and Raspberry Pi (RP)

Scheme	Computation Cost Equation	CC on Desktop	CC on RP4	EC on Desktop	EC on RP4
[101]	$4T_{ECC}+5T_{sign}+5T_{veri}$	0.892	35.4	9.705	84.96
[102]	$3T_{ECC}+6T_{sign}+6T_{veri}$	0.975	39.6	10.608	95.04
[108]	$16T_{H(.)}+5T_{ECC}+4T_{sign}+4T_{veri}$	0.822	31.28	8.94	75.07
[146]	$6T_{H(.)}+2T_{ECC}+4T_{sign}+4T_{veri}$	0.655	26.43	7.13	63.43
[147]	$8T_{H(.)}+5T_{sign}+5T_{veri}$	0.686	29.0	7.47	69.69
BLISS	$7T_{H(.)}+4T_{sign}+4T_{veri}$	0.550	23.23	5.99	55.75
SCAB-IoTA	$5T_{H(.)}+5T_{ECC}+2T_{AES}$	0.519	19.95	5.65	47.88

Storage and Communication Cost

Storage cost refers to the storage required for storing the required values of multiple variables used in the execution of the scheme, such as the secret key, hash code, and other parameters in all phases. On the other hand, communication cost refers to the storage consumed by the scheme based on the used variables for communication, such as hash code, ciphertext, and other parameters. The storage and communication cost is measured in Bytes. We have derived the equations for BLISS, SCAB-IoTA, and state-of-the-art schemes to compute the storage cost and communication cost shown in Table 5.4. The required storage for different cryptographic functions used in the schemes is as follows. A secret Key S_K requires 24B, ECC encryption/decryption S_{ECC} takes 48B, AES encryption/decryption S_{AES} takes 24B, a hash function $S_{H(.)}$ requires 32B, and ECDSA based signature S_{sign} and verification S_{veri} requires 10B. Table 5.4 represents the comparative storage cost of the BLISS, SCAB-IoTA with state-of-the-art schemes.

Table 5.4: Comparative storage and communication cost of BLISS, SCAB-IoTA with state-of-the-art schemes

Scheme	Storage Cost	Cost in Bytes	Communication Cost	Cost in Bytes
[101]	$10S_{H(.)}+1S_{ECC}+1S_{sign}$	378	$7S_{H(.)}+2S_{ECC}+2S_{sign}$	340
[102]	$4S_{H(.)}+2S_{ECC}+2S_{sign}+2T_k$	332	$2S_{H(.)}+2S_{sign}+2T_k$	212
[108]	$3T_{H(.)}+4S_{sign}+4T_k$	376	$3S_{sign}+2T_k+ConstantVar$	242
[147]	$3S_{H(.)}+2S_{sign}$	116	$4S_{H(.)}+2S_{sign}$	148
BLISS	$2S_{H(.)}+2S_{sign}$	74	$2S_{H(.)}+4S_{sign}$	94
SCAB-IoTA	$4S_{H(.)}+1S_{AES}+3S_{ECC}$	296	$1S_{H(.)}+2S_{AES}+3S_{ECC}$	224

5.1.7 Security Analysis

Here, we briefly explain how both the proposed schemes, BLISS and SCAB-IoTA, withstand multiple attacks. We have also defined how SCAB-IoTA ensures the authentication and integrity property of the security. Table 5.5 shows a comparative analysis of BLISS and SCAB-IoTA with state-of-the-art schemes. In this table, ✓means the scheme is protected, and × means the scheme is not protected. The proof description of all the attacks is defined below.

Lemma 9 (Authentication)**.** *BLISS secures the authentication of IoT devices and the integrity of data exchanges by employing digital signatures, specifically utilizing ECDSA and SHA256 algorithms. Under the chosen-message attacks, the probability of successfully forging a signature is negligible for any adversary* $\mathbb{A}$*. This is formalized by the conditions that for a given key pair* (C^i_{pub}, C^i_{pri}) *generated by* $KEYGEN(k)$*, and for any message* M *not in the set of signed messages* $M_1, M_2, \ldots, M_n$*, the verification function* $VERIFY(C^i_{pub}, M, \sigma) = True$ *implies a negligible success probability for* $\mathbb{A}$*, assuming* σ_M *is a signature on* M *that* $\mathbb{A}$ *computes without knowing* C^i_{pri}*. Thus, BLISS ensures a robust authentication protocol for IoT ecosystems.*

Table 5.5: Comparative security analysis of BLISS and SCAB-IoTA with state-of-the-art schemes

Types of attacks	[101]	[102]	[104]	[108]	**BLISS**	**SCAB-IoTA**
Identification	✓	✓	✓	✓	✓	✓
Message Integrity	✓	✓	✓	✓	✓	✓
Sybil Attack	✓	✓	✓	✓	✓	✓
Fake Node Injection	✓	✓	✓	✓	✓	✓
Tampering	✓	×	✓	×	✓	✓
Message Replay Attack	×	✓	✓	×	✓	✓
Man-in-Middle Attack	✓	✓	✓	✓	✓	✓
Spoofing Attack	✓	✓	✓	✓	✓	✓

Proof. We have used the digital signature based on the ECDSA and SHA256 because it is unalterable under any random attack. Let the message space be M, and the adversary is $\mathbb{A}$. The success possibility of adversary $\mathbb{A}$ is minimum or almost negligible if it satisfies the following computation.

Let $M_1, M_2, ...M_n$ are the set of messages chosen randomly from message space M and key-pair for candidate C_i (C^i_{pub}, C^i_{pri}) is generated using a function called $KEYGEN(k)$. Signature for the set of randomly chosen messages can be computed as $\sigma_{M_1} = SIGN(C^i_{pri}, M_1), ...,\ \sigma_{M_n} = SIGN(C^i_{pri}, M_n)$. $\mathbb{A}$ is given C^i_{pub} and $(M_i, \sigma_i)^n_{i=1}$ and output (M, σ_M). The success of $\mathbb{A}$ is possible if $VERIFY(C^i_{pub}, X, \sigma_X) = True$ and $X \notin (M_1, M_2, ..., M_n)$. However, the BLISS is strongly unalterable if this step is modified according to the following step. $VERIFY(C1_{pub}, X, \sigma_X) = True$ and $(X, \sigma_X) \notin (M_1, \sigma_{M_1}), (M_2, \sigma_{M_2}), ..., (M_n, \sigma_{M_n})$.

$\mathbb{A}$ could observe the communication channel as it is open. However, messages (M_n) from the C_i to C_j or from C_i to controller are signed $\sigma_{M_n} = SIGN(C^i_{pri}, M_n)$ and $VERIFY(C^i_{pub}, M_n, \sigma_{M_n})$ verified by C_j or controller before any further processing of the message. the $\mathbb{A}$ cannot reverse the transaction's hash, and it does not have the private key of the candidate or controller. Even if the $\mathbb{A}$ seizes the channel and obtains the signed messages, then fake IoT device authentication is impossible. □

Lemma 10 (Integrity)**.** *The BLISS ensures the data integrity of the shared messages between the candidate and controller.*

Proof. Let the message space be M, and the adversary is $\mathbb{A}$. The success possibility of adversary $\mathbb{A}$ to violate the integrity of the shared message is minimal or almost negligible if it satisfies the following computation. Let $M_1, M_2, ...M_n$ are the messages generated by $\mathbb{A}$, and it also know the public key C^i_{pub} of candidate C_i. The $\mathbb{A}$ somehow know signature for the set of randomly chosen messages $\sigma_{M_1} = SIGN(C^i_{pri}, M_1), ...,$ $\sigma_{M_n} = SIGN(C^i_{pri}, M_n)$. $\mathbb{A}$ is given C^i_{pub} and $(\sigma_{M_i})^n_{i=1}$ and output (M, σ_M). The success of $\mathbb{A}$ is possible if $VERIFY(C^i_{pub}, X,\ \sigma_X) = True$ and $X \notin (M_1, M_2, ..., M_n)$. However, the BLISS is strongly unalterable if this step is updated according to the following step. $VERIFY(C^i_{pub}, X, \sigma_X) = True$ and $(X, \sigma_X) \notin (M_1, \sigma_{M_1}), (M_2, \sigma_{M_2}), ..., (M_n, \sigma_{M_n})$.

BLISS uses signature and verification algorithms. Suppose a third party has access to the transactions and attempts to falsify them. It is only C_j who can decipher and verify M_1, a message signed with C_i's private key, which is sent to C_j. Private keys of both candidates and controllers can not be obtained by an $\mathbb{A}$. For example, if another candidate or controller's private key is needed, it is impossible for $\mathbb{A}$ to sign this transaction. Because of this, the system ensures the integrity of the transactions and the blockchain's immutability and tamper-proof record of the information. □

A. Identity Theft

Identity theft would necessitate an adversary $\mathbb{A}$ to steal the secret information of a candidate device. In BLISS and SCAB-IoTA, all secret information is stored on the blockchain in the form of hash codes, providing anonymity and preventing $\mathbb{A}$ from obtaining the actual identity of any device due to the computational infeasibility of reversing hash functions.

- **In BLISS:** Authentication is handled by the controller issuing a $Sign^i_{TKN}$, which is a digitally signed of C^i_{addr} and hash of the cluster ID and timestamp using the controller's private key Ctr_{pri}. For successful communication, a device must present $Sign^i_{TKN}$, its address C^i_{addr}, and an authentication code $Auth^i_{code}$, which are validated prior to communication. Assuming $\mathbb{A}$ attempts to impersonate a candidate C_j to send a forged message to C_i, it would require C^i_{addr} and C^j_{pri}. However, C^j_{pri} is confidential and $\mathbb{A}$'s requests for C^i_{addr} must be accompanied by a valid $Sign^i_{TKN}$, which $\mathbb{A}$ cannot fabricate without access to Ctr_{pri}. The controller's vigilant monitoring for anomalous behavior and the necessity of a valid $Sign^i_{TKN}$ for each request enables it to discard any attempts at identity theft by adversaries effectively. Therefore, the BLISS scheme's reliance on secure hash functions and strict authentication protocols ensures its security against identity theft.
- **In SCAB-IoTA:** Each member of the cluster is identified by the cluster head, which assigns its public key Ctr_{pub} for future communication as $\kappa = Ctr_{pub} \oplus n_2$, $\xi = D_{ji} \oplus n_2$, and the device which receives response do computation as $Get(D_{ji}) = \xi \oplus n_1 \oplus C_i$, $Get(Ctr_{pub}) = \kappa \oplus n_2$ and get the public key of the controller i.e., only the identified member can communicate to other members as well as from controller. Along with the public key, the controller generates the one-time secret key S_K for M2M communication, which is only assigned to identified members. Hence, our SCAB-IoTA withstands identification attacks.

B. Message Integrity

It is a kind of attack when the integrity principle of security is violated, i.e., when someone can modify the content of the message in transit. The integrity of the message is retained because the interaction between members is secured with a one-time secret key, i.e., encryption $E[S_K[Msg1]]$ and decryption $D[S_K[Msg1]]$, and interaction between the candidate and controller is secured with public-key encryption $E[Ctr_{pub}[Msg1]]$. The controller generates a one-time secret, so there is no secret key compromising.

C. Sybil Attack

In the face of Sybil attacks, where an adversary may attempt to subvert the network by creating multiple fake identities, the BLISS and SCAB-IoTA scheme enforces a strong authentication process.

- **In BLISS:** Each device within the network is assigned a unique set of credentials that include D_{ji}, C^i_{addr}, $Auth^i_{code}$, and C^i_{pub}. These credentials are securely recorded and maintained by the controller in each block. Smart contract rules are applied rigorously before allowing a device to join the cluster or to enable inter-device communication. These contracts, backed by lightweight cryptographic functions, are designed to authenticate each device's credentials, ensuring that each identity is verified and authenticated by the controller. This systematic approach to authentication, coupled with the security

provided by the blockchain ledger, renders the possibility of forging identities and executing a Sybil attack infeasible. The controller's role in uniquely identifying and verifying each cluster member constitutes a robust barrier against the threat posed by Sybil attacks. Hence, the BLISS scheme presents strong protection against Sybil attacks.

- **In SCAB-IoTA:** Each member device ID is stored at blockchain node $block = [D_{ji}, Req_i, T_3, BH, PBH]$, $CT = E(BK_{pub}(block))$. Hence, the device can have only one identity, and each member can have only one secret key at a given time for communication. All communicating messages are secured by the secret key S_K associated with this member ID D_{ji}. Moreover, all the participating members must be authenticated by the controller. Thus, an attacker cannot use fake identities.

D. Fake Node Injection Attack

A fake node injection attack in a blockchain-based IoT system occurs when an attacker introduces a malicious node into the network, aiming to disrupt the data flow or gain unauthorized access to sensitive information. The BLISS is designed to mitigate such security threats by ensuring that all participating nodes are authenticated and verified before they can communicate within the network. BLISS ensures that only authenticated candidate devices can join and communicate within a cluster, thus preventing fake node injection attacks.

The C^i_{pub} and C^i_{pri} is the public-private key pair of a candidate generated using ECDSA. The C^i_{addr} is the unique address of the candidate device, computed as $C^i_{addr} = H(C^i_{pub} \oplus F_n)$. The authentication code $Auth^i_{code}$ is given by $Auth^i_{code} = H(C^i_{addr} \oplus n)$, ensuring uniqueness due to the pre-image resistance of H. In the cluster creation phase, the controller issues $Sign^i_{TKN} = SIGN[Ctr_{pri}[C^i_{addr} H[Cst_{ID}||T_2]]]$. As the SIGN operation is secured by the private key Ctr_{pri}, $Sign^i_{TKN}$ cannot be replicated by an adversary. Furthermore, the smart contract mandates a valid $Sign^i_{TKN}$ for a candidate to join the cluster, verified by $b \leftarrow VERIFY(Ctr_{pub}, Sign^i_{TKN}, \sigma_{Sign^i_{TKN}})$. A fake node cannot participate in the cluster's communication without the correct $Sign^i_{TKN}$. Thus, by the properties of hash functions, ECDSA, and digital signatures, the BLISS scheme is provably secure against fake node injection attacks.

E. Tampering Attack

A *tampering attack* in a blockchain-based IoT system context refers to any malicious activity where an adversary intentionally alters, manipulates, or substitutes the data or devices within the system. This type of attack can manifest in various forms:

1. Altering the data stored on the blockchain to falsify records.
2. Physically tampering with IoT devices to change their behavior or output.
3. Manipulating the communication between nodes to inject or alter data.
4. Substituting legitimate devices with compromised ones to gain unauthorized access or disrupt the system's normal operation.

This alteration aims to disrupt the system's normal operation, introduce fraudulent data, or gain unauthorized access to system functionalities. Formally, a tampering attack is a function $T : N \rightarrow N'$, where N is the

set of legitimate nodes, and N' is the set of compromised nodes, such that the altered node's behavior or reported data deviates from the expected or genuine behavior. The proposed scheme BLISS ensures that it is protected from this attack. Below, we have described how BLISS is secured from tampering attacks. The authentication code $Auth^i_{\text{code}}$ is unique and securely bound to the device identity and nonce. The $Sign^i_{\text{TKN}}$ is a token signed by the controller's private key, which is unique to the controller and cannot be forged. The BLISS scheme ensures that every communication in the network is signed and verified using the public-private key pairs of the devices and controller. The integrity provided by ECDSA for signatures means that any modification of the data or the node identity will result in a signature verification failure, thus detecting any tampered message or device. Therefore, for all data d_i and nodes n_i, a verification function $\text{VERIFY}(C1_{\text{pub}}, d_i)$ returns false for tampered d_i. Authentication is guaranteed since each candidate device generates its own keys and unique address, and the controller verifies these during the join request, allowing only authenticated devices to join the network. Thus, a device C_i is authenticated if and only if $Sign^i_{\text{TKN}}$ is valid. Moreover, the signed token $Sign^i_{\text{TKN}}$ ensures non-repudiation, establishing that only the controller with the private key could have created it, making any message or command from the controller trusted and irrefutable. Hence, a verification function $VERIFY(Ctr_{\text{pub}}, Sign^i_{\text{TKN}})$ returns true if $Sign^i_{\text{TKN}}$ is valid. These cryptographic checks and balances effectively prevent tampering attacks since the system detects and rejects any unauthorized modifications to device identities, data, or communication.

F. Message Replay Attack

Both the BLISS and SCAB-IoTA schemes employ a unique transaction identifier and synchronized timestamps to prevent replay attacks. The authentication and cluster creation phase details how a request is formed as $Req[C^i_{addr}||D_{ji}||Sign^i_{TKN}||\sigma_{M_{s1}}||T_1]$, embedding the candidate's unique address C^i_{addr}, an identity D_{ji}, and a timestamp T_1 within each message. The timestamp T_1 ensures that each request is inherently tied to a specific point in time, enabling the network to distinguish between new and replayed messages. A message is considered valid only if the current network time T_{curr} minus T_1 does not exceed the defined interval ΔT. This protocol ensures that any message with a timestamp that does not conform to the temporal constraints is rejected, thereby preventing replay attacks. Moreover, the blockchain ledger records all transactions and, through network consensus, makes any replayed action detectable and preventable. Once an action is confirmed and recorded, it cannot be replayed, contributing to the security of the network against replay attacks. Therefore, by combining the unique message identifiers, the inclusion of synchronized timestamps, and the blockchain's immutability, the BLISS and SCAB-IoTA schemes ensure comprehensive protection against replay attacks, maintaining the integrity and security of the network's communications.

G. Man-in-The-Middle Attack

In the BLISS scheme, communication between devices and the controller, or between devices themselves, is secured using cryptographic signatures. Assume an adversary $\mathbb{A}$ has intercepted the communication and obtained the public key $C1_{\text{pub}}$ and address $C1_{\text{addr}}$ of a candidate device C_1. In the BLISS scheme, messages between devices or between a device and the controller are signed with the sender's private key, which $\mathbb{A}$ does not possess, thus preventing message forgery.

Furthermore, authentication in BLISS requires presenting a valid signed token $Sign^i_{\text{TKN}}$ to the controller, which is signed using the controller's private key and can only be verified with the public key Ctr_{pub}. Without the capability to produce a valid $Sign^i_{\text{TKN}}$, $\mathbb{A}$ cannot impersonate a legitimate device. Additionally, the integrity of messages is protected by including the timestamp T_2 in the message composition. This ensures

that even if $\mathbb{A}$ intercepts a message without the correct private key to sign a forged message, the network will reject the communication due to failed signature verification. Therefore, the BLISS scheme, through its use of digital signatures, secure authentication process, and timestamp verification, is protected against man-in-the-middle attacks, maintaining the confidentiality and integrity of the network's communications.

H. Spoofing Attack

In IoT systems, a *spoofing attack* occurs when an attacker disguises as a legitimate entity to gain unauthorized access to the system's resources or sensitive information. This involves the attacker presenting false credentials or manipulating communication protocols to mislead the system into accepting an illegitimate entity as authentic.

- **In BLISS:** For a spoofing attack to be successful, an attacker must forge $Sign^{i}_{\text{TKN}}$ to impersonate a legitimate device. Due to the collision resistance property of the hash function H, finding such an x that $H(x) = Sign^{i}_{\text{TKN}}$ is computationally infeasible. Furthermore, without the corresponding C^{i}_{pri}, no attacker can generate a valid signature that the BLISS controller would accept because $VERIFY(Ctr_{\text{pub}}, Sign^{i}_{\text{TKN}}) =$ true, if and only if $Sign^{i}_{\text{TKN}}$ was created with Ctr_{pri}. This is due to the security properties of the digital signature. Hence, the BLISS scheme, by the cryptographic strength of ECDSA and the hash function H, ensures that spoofing attacks are infeasible.
- **In SCAB-IoTA** Unlike the Sybil attack, the attacker spoofs the identity of a recognized device to gain access to his/her rights. As we have already mentioned in SCAB-IoTA, the controller does the authentication of participating devices and distributes the key for communication, i.e., send(κ and ξ), where $\kappa \leftarrow Ctr_{pub} \bigoplus n_2$, and $\xi \leftarrow D_{ji} \bigoplus n_2$. Hence, an attacker cannot spoof another member's identity since he needs to be authenticated by the controller.

5.2 SOLUTION FOR ITS APPLICATIONS

Road accidents are the 9^{th} biggest cause of death, which figures around 1.5 million people per annum globally [44]. 1.5 lakh people were killed, and 5 lakh people were injured in India, according to the road accident database report 2019-20 [45], [46]. It is found in the studies that the countries that do not invest in road safety could lose about 7 to 22% in potential per capita GDP growth in the coming years. In addition to life and economic loss, traffic jams cause the loss of fuel, time, and pollution. Therefore, any country needs to make decisions for road safety policies. The Intelligent Transportation System (ITS) is the right answer, and it is a rapidly growing concept in the field of computer science [148]. ITS aims to achieve road traffic efficiency by reducing road traffic-related problems. It informs users in advance with prior information about road traffic jams, accident zone areas, slow and fast speed zones, local convenience real-time running information. Moreover, it helps reduce travel time and enhance safety and comfort. The ITS aims to provide road safety and efficiency in terms of reducing time, cost, and pollution [149]. In ITS, blockchain technology has been utilized to enhance security and efficiency. Blockchain provides a secure communication framework for ITS applications, protecting against various cyber attacks, including man-in-the-middle attacks, impersonation attacks, device tampering, and Sybil attacks. The use of TPD (Tamper-Proof Devices) in conjunction with blockchain and SDN (Software-Defined Networking) ensures that critical information, such as vehicle identities and secret messages, cannot be tampered with. Implementing blockchain in ITS

transforms it into a decentralized application, which further reduces the risk of a single point of failure. In addition to decentralization, the secure framework and untampered information provided by blockchain enhance transparency in ITS, thereby increasing trust among participants in the transportation ecosystem, including vehicles, infrastructure, and service providers. Similar to IoT applications, the overhead introduced by blockchain, in terms of latency and low throughput, directly impacts the performance of ITS, where real-time information exchange and low-latency communication are critical. Moreover, integrating blockchain with ITS adds complexity to the system, particularly in terms of operational challenges. Therefore, it is necessary to design a suitable framework or protocol based on blockchain for ITS that addresses these integration issues, which includes the development of an appropriate consensus algorithm.

We have developed two schemes for ITS applications using blockchain technology. The first scheme is LBSV, a lightweight blockchain security protocol for secure storage and communication in SDN-enabled Internet of Vehicles (IoV), and the second is SmartCoin, a novel incentive mechanism for vehicles in ITS based on consortium blockchain. While distinct, these systems collectively represent blockchain's innovative use in ITS. LBSV and SmartCoin collectively showcase blockchain technology's potential to enhance the security and efficiency of ITS. While LBSV focuses on securing the IoV through SDN and blockchain integration, SmartCoin incentivizes honest vehicles, promoting a reliable and efficient ITS environment.

5.2.1 Motivation and Contribution

The fundamental issue faced by ITS is the use of a traditional centralized approach to provide security and privacy in the network, which is prone to a single point of failure. Indeed, secure communication depends on efficiency and the applied technique's fairness. Satisfying ITS network requirements, along with low computational cost, is another challenging problem faced by the research community. In LBSV and SmartCoin, we investigate a lightweight blockchain security protocol using SDN-based architecture to improve confidentiality and secure message dissemination. Secure message dissemination issues have been explored with the following question: *"How to devise a technique that can securely deliver a message in an untrustworthy ITS environment?"*. Centralized controlling issues have been explored to answer the question *"how to create a distributed and decentralized framework for ITS using blockchain technology"*. Further, this research is motivated by the challenge, *"how to improve the computational efficiency of blockchain and use a modified consensus algorithm to achieve secure communication"*. Moreover, it is quite challenging to handle storage requirements in the network. So, we need to explore *how to create lightweight blocks and small consensus mechanisms to overcome storage requirements*. ITS communication uses a tremendous amount of data and controls messages for the coordination and cooperation of vehicles. Specifically, the contributions of our proposed scheme are as follows:

- We have proposed a modified Practical Byzantine Fault Tolerance (mPBFT) consensus algorithm suitable for permissioned blockchain for SDN-enabled IoV network in LBSV. The mPBFT consensus scheme ensures fairness, i.e., it gives equal chances to all the miners to propose a block. An improved leader selection technique has been used to ensure fairness in the mPBFT consensus algorithm using a modular function.
- A hybrid SDN-based architecture is used in LBSV to reduce the controller's overhead and provide a distributed environment. Secure message dissemination is achieved through cryptographic hash functions and cryptographic digital signature techniques.

- To improve the network efficiency and maximum honest public participation, a novel incentive mechanism based on consortium blockchain is proposed in SmartCoin. A new digital cryptocurrency has been introduced and used as an incentive for vehicles called SmartCoins (STC); due to this, we also name the whole scheme as SmartCoin.
- SmartCoin also employed a proposed lightweight consensus algorithm (LCA) for transaction confirmation. SmartCoin provides the authentication of the participating vehicles and integrity and confidentiality of the vehicle information and the messages communicated.
- To show the efficiency of the LBSV and SmartCoin, the performance analysis is performed based on the computation cost, energy consumption, storage cost, and communication cost. We also provide the security analysis to show how the proposed scheme is resistant to various cyber-attacks, e.g., impersonation, botnet, Sybil, man-in-the-middle, message replay, and modification. We have also presented a lemma-proof of the security of Smartcoin for authentication, data integrity, double-spending, forking attacks, and fault tolerance.

5.2.2 Network Model

There are various components in the proposed mechanisms, i.e., vehicles, roadside units (RSU), controllers, and tamper-proof devices. As shown in Figure 5.9, RSUs and vehicles can communicate with each other using broadcast over the public channel using Dedicated Short Range Communication (DSRC) [150] protocol. We have divided the architecture into two layers. The consensus layer consists of SDN controllers and blockchain storage. Storing complete data in blockchain creates storage issues; we have used blockchain only to store tamper-proof data and create lightweight blocks. Each vehicle in the IoV layer needs to authenticate itself before involving in communication. The authentication process is carried out through the controller using a secure channel (e.g., Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocol). Each vehicle is equipped with a Tamper Proof Device (TPD). Table 5.6 depicts the mathematical symbols that we have used in the methodology of both schemes. Below, we defined each component of the ITS network model.

- **Tamper Proof Device:** TPD is a specialized hardware component embedded in vehicles to securely store and manage sensitive cryptographic keys and other critical security parameters. These devices are designed to resist physical tampering, ensuring that even if an attacker gains physical access to the vehicle, they cannot extract or manipulate the stored keys. In the context of Intelligent Transportation Systems (ITS), TPDs play a crucial role in establishing secure communication channels between vehicles and infrastructure by safeguarding the integrity and confidentiality of cryptographic keys used for encryption, decryption, and digital signatures. A TPD is installed in the vehicle as a computational unit. It stores the vehicle's ID ID_v^i, user's ID ID_u^i, and nonce n_u. The TPD does the initial computation for the initialization phase. The hash (SHA256) is used to compute the hash code for the registration and communication phase. After the initialization phase, it generates the public and private key pair (V_{pub}^i, V_{pri}^i), which is used for the ECC encryption/decryption. It also stores the IDs ($ID_u^i, ID_v^i, ID_{RSU}, ID_c$) and a list of public keys of other vehicles Ls_k. Figure 5.10 illustrates the TPD with essential computations.

  While TPDs are crucial for safeguarding cryptographic keys and preventing unauthorized access at the device level, it is important to note that they cannot fully mitigate network-level threats such as man-in-the-middle attacks, replay attacks, or message tampering. These vulnerabilities highlight the

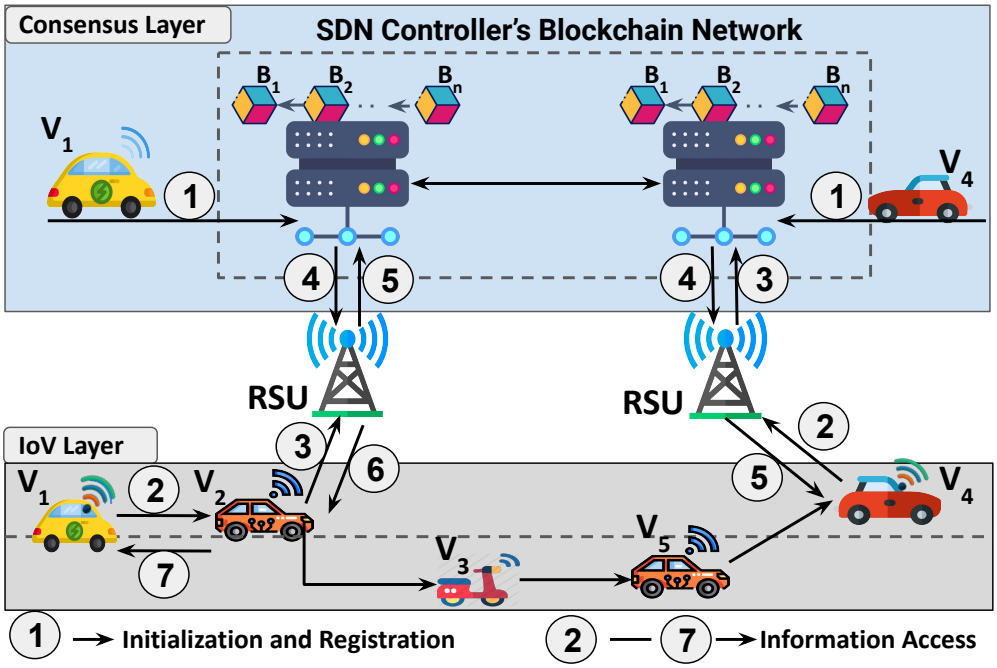


Figure 5.9: A basic network model of proposed ITS solution

necessity for additional security measures, such as those provided by the LBSV scheme, which are designed to secure communications across the network.

- **Vehicle:** A vehicle is the only mobile unit in the ITS network. It is equipped with TPD for communication with other components of the network, such as vehicles, RSUs, and controllers.
- **Roadside Unit (RSU):** The RSU is the first point of contact for the vehicle to interact with the controllers in the communication phase. Further, it is responsible for broadcasting updated information in the ITS network. RSUs also work as a bridge between the controller and vehicles. RSU exchanges network topology and other related information with the SDN controller. The data plane in SDN is used to manage both RSUs and vehicles.
- **Controller:** It is the high-processing entity of the ITS network and a trusted component. The SDN controller is connected to multiple controllers, and only the controller can add blocks in the blockchain and distribute them amongst other controllers. Controllers use a modified Practical Byzantine Fault Tolerance (mPBFT) consensus algorithm in LBSV and a lightweight consensus algorithm (LCA) in SmartCoin to run the blockchain functionalities.
- **SDN Operations:** SDN architecture provides a multitenant environment and offers flexibility to programmers to manage the increasing complexity of computing systems. In the context of ITS, heterogeneous communication introduces several security challenges that can lead to information leakage. By using SDN, we can separate application services from security services, thereby enhancing information security. The integration of blockchain technology with SDN can address the single point of failure typically associated with centralized systems, creating a decentralized computing environment. SDN architecture also offers a global overview of the network and facilitates management services such as protocol enabling, topological information, and communication setup.

Our proposed scheme utilizes the OpenFlow protocol for communication. LBSV incorporates a hybrid structure to facilitate communication between different components. The SDN controller is responsible for controlling network behavior and creating the blockchain. We assume that all vehicles are required to send information exchange messages to the control plane in SDN. The control plane then evaluates the optimum route to the destination as requested by the source vehicle. Each vehicle prepares its routing table based on the algorithm running at the controller plane. The controller recalculates the routing paths if an information packet fails to reach its destination. If consecutive failures occur for the same request, the route will be considered a failure. In the hybrid scenario, SDN does not bear all the communication overhead; instead, local computations are performed by the RSUs and the vehicles. The TPD in the vehicle functions as an OpenFlow switch, while the RSUs are used to exchange topological information with the SDN. Our configuration has three primary entities: the SDN controller, TPDs functioning as OpenFlow switches, and the SDN RSUs.

While SDN introduces centralized control for the efficient management of network operations, the LBSV scheme employs multiple SDN controllers to create a decentralized blockchain network. This decentralized approach eliminates any single point of failure, thereby enhancing the system's resilience. These SDN controllers collaborate to maintain the blockchain, ensuring secure communication and the transmission and authentication of messages across the network. The integration of SDN with blockchain technology, along with the use of multiple controllers, also allows for dynamic management of network resources, making the LBSV architecture well-suited to the dynamic and complex environment of ITS.

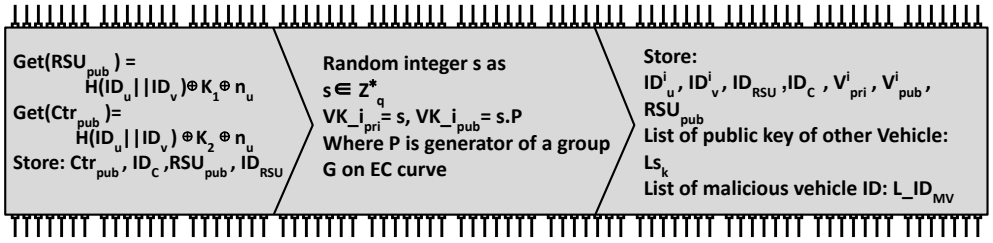


Figure 5.10: Representation of TPD with computation of different parameters.

Table 5.6: Symbol table for LBSV and SmartCoin

Symbol	Description	Symbol	Description
σ_x	Digital Signature of x	$H(x)$	Hash (SHA256) function for the input x
V_i	i^{th} Vehicle	ID_u^i	ID of i^{th} user
ID_v^i	ID of i^{th} vehicle	ID_{RSU}	ID of i^{th} RSU
ID_c	ID of i^{th} controller	Ctr_{id}	Controller's ID
$\triangle T$	Threshold time	V_{pub}^i, V_{pri}^i	Public-private Key-pair of vehicle $\forall$ i = 1,2,...
Ctr_{pub}, Ctr_{pri}	Public-private Key-pair of controller	RSU_{pub}, RSU_{pri}	Public-private Key-pair of RSU
Tr_1, Tr_2,....	Transactions	$Mrt()$	A function representation to create Merkle root hash

5.2.3 Assumptions

We have made some assumptions in the scheme. Before being part of the ITS network, a basic setup for the smart vehicle is needed. This setup is required for the necessary credentials for registration and communication. The manufacturing factory and user should perform some computation procedures.

1. The dedicated short-range communication (DSRC) protocol has been amended and renamed as wireless access in vehicular environments (WAVE). The WAVE protocol uses the IEEE 802.11p wireless communication standard. According to the standard, every vehicle disseminates messages related to road traffic in the time interval of 100–300 ms to other vehicles or RSUs. According to WAVE protocol, smart transportation's communication rate ranges from 6 to 27 Mbps, with the maximum broadcast coverage ranging up to 1,000 m.

2. A user U_i gets ID_v^i after purchasing a smart vehicle. Then U_i generates some random nonce n_u and computes $X_i = U_i \oplus ID_v^i \oplus n_u$ and sends $< U_i, X_i >$ to the factory through the secure channel [SSL/TLS].

3. The factory already knows the ID_v^i corresponding to the U_i. The factory receives the request, extracts the random nonce as $n_u = X_i \oplus ID_v^i \oplus U_i$, and computes $Y_i = H(X_i||n_u||n_f)$ with random nonce n_f. Further, The factory stores the Y_i in its secure storage. We assume that the factory database is highly secure. The factory also stores Y_i in the TPD and sends it to the user to install it in the vehicle. The factory's database is shared with the controller node of the ITS network to identify the authentic vehicles.

4. After installing the TPD in the vehicle ID_v^i, it generates the keypairs (V_{pub}^i, V_{pri}^i) using the ECC cryptographic algorithm. We also assume that TPD can calculate the hash (SHA256), digital signature, and verification of the messages.

5. The RSUs and controller node also generate their keypair (RSU_{pub}, RSU_{pri}) and (Ctr_{pub}, Ctr_{pri}) respectively using the ECC, and they share their public key with other vehicles and RSUs.

5.2.4 Adversary Model

We have used widely accepted Canetti and Krawczyk's (CK) adversary model for the proposed scheme [151]. The adversary model lists the assumption that is considered from the ITS network model. We assume that each vehicle is equipped with TPD and can communicate with RSUs and other vehicles in the range of TPD.

- Any vehicle/RSU/controller that acts like the adversary can send a false transaction/message on behalf of any legitimate vehicle/RSU/controller if it has all the necessary credentials (during initialization, registration, and communication phase).
- The adversary should know the private data to compromise security. To satisfy this condition, the calculated and received hashes of the intermediate variables have to match.
- A legitimate entity would act as an attacker if it tried to impersonate another entity using its credentials. Blockchain could be beneficial in this scenario to overcome impersonation along with the ECC and SHA256 cryptographic functions.
- A TPD is a tamper-proof physical device that stores the encrypted information and decrypts it using the user's private key using the ECC cryptographic technique. Thus, an adversary can not extract any information from TPD.
- Due to public channel message communication, the adversary can interrupt/modify/stop/delay the message dissemination and perform other malicious activities. The message authentication code and ECC provide authentication and integrity.

5.2.5 LBSV: Lightweight Blockchain Security Protocol for Secure Storage and Communication in SDN-enabled IoV

LBSV is a security protocol for storage and communication in IoV. IoV is a critical element of the ITS network, which has many modes of communication, such as vehicle-to-vehicle (V2V), vehicle-to-RSU (V2R), or infrastructure (V2I), RSU-to-SDN controller (R2C), and vehicle-to-an SDN controller (V2C). We have considered a two-layered network model in LBSV as shown in Figure 5.9, which are the IoV layer and consensus layer. The IoV layer comprises vehicles and RSU, and the consensus layer comprises SDN controllers. The SDN controllers create the blockchain network and follow the modified Practical Byzantine fault Tolerance (mPBFT) consensus *Algorithm* 14 for message verification and block generation. All the entities of the IoV network have the keypair for secure communication. Many blockchain-based solutions have been proposed for IoV networks to provide secure information storage and communication. However, monitoring the network effectively and focusing on the integration of SDN architecture [152] with blockchain technology is essential. Differentiating from the state-of-the-art techniques, LBSV tackles security-related issues such as secure storage, communication, and efficient computation. The conventional security solutions are not directly applicable to IoV architectures due to their high computation and storage demand. Therefore, we introduce a blockchain-based architecture to provide a security solution for ITS applications. The distributed nature of blockchain provides leak-proof security and ensures tamper-proof data. LBSV uses an mPBFT consensus algorithm to overcome the limitation in terms of consensus delay present in traditional consensus algorithms such as conventional PBFT, PoW, and PoS. Further, for the resource constraint device in IoV, we used lightweight cryptographic functions such as ECC and the SHA256 for secure communication, authentication, and data integrity. Further, The SDN controller is programmed to operate in a hybrid manner and provide a global network view. The SDN controller does not assert full control over the network. Instead, it communicates with the RSUs to exchange routing policies. Resource availability and routing control are also managed by the control plane of SDN [153]–[155].

LBSV consists of three phases, which are the initialization phase, the registration phase, and the communication phase. In the initialization phase, a vehicle interacts with the SDN controller for the registration process. This is the first requirement for a vehicle to be a part of the IoV. The vehicle communicates with the SDN controller over the secure channel and exchanges credentials for the next phase, i.e., registration. The vehicle generates its public-private key pair and gets the TPD in the registration phase. The SDN controller also creates a block for the vehicle registration and shares it with the other SDN controller of the network. Notably, in the registration phase, the vehicle gets its TPD for communication and registers its public key to distribute in the network. The initialization and registration phases are necessary to protect the IoV network from various attacks. Along with protection from attack, it also provides confidentiality, integrity, and authentication for the communication between vehicles, RSUs, and SDN controllers. After the successful registration phase, the vehicle is ready to be a part of the IoV layer and communicate with RSUs and other vehicles in the IoV network. In the communication phase, V2V communication takes place. The V2V communication could be direct communication or via RSU, as depicted in Figure 5.9. The detailed description of all these phases is described in the subsequent sections.

Initialization Phase

Every vehicle goes through an initialization phase before starting any communication with the entities. The initialization is carried out between the vehicle and the controller. Here, we assume the controller has a

public key and RSU IDs. As shown in Figure 5.11a shows the initialization process of the proposed scheme. The steps of the initialization process are as follows:

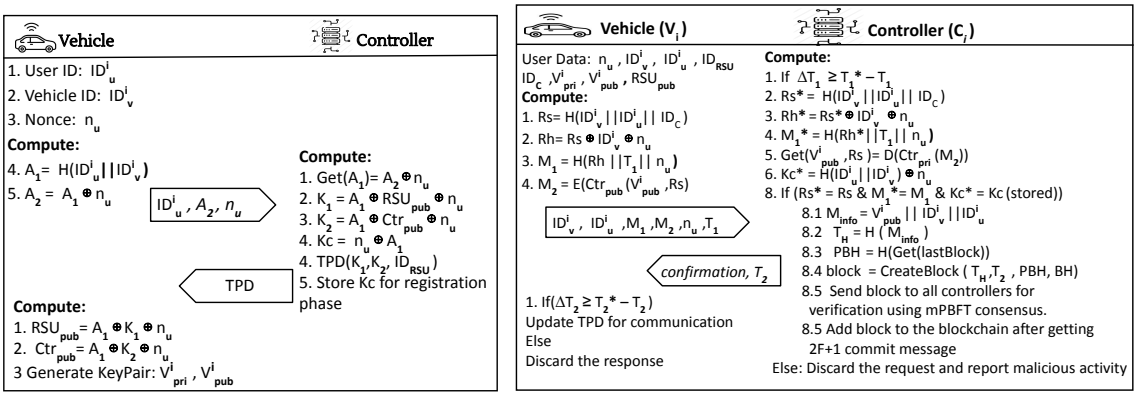
1. The user has its user ID, ID_u^i and vehicle ID, ID_v^i. It chooses a pseudo-random number called nonce n_u. Then it compute A_1 as $H(ID_u^i||ID_v^i)$ and A_2 as $A_1 \oplus n_u$. After that, the user sends $< ID_u^i, A_2, n_u >$ to the controller using a secure channel for getting the confidential credentials for future communication.

2. On receiving a request from the user, the controller computes A_1 from the received data as $A_2 \oplus n_u$. After computing A_1, it calculates K_1, K_2 and Kc as $A_1 \oplus RSU_{pub} \oplus n_u$, $A_1 \oplus Ctr_{pub} \oplus n_u$ and $n_u \oplus A_1$ respectively. Then, after storing the value of K and ID_{RSU} into the TPD, the controller also stores the value of Kc in its local repository against the vehicle ID for the registration phase.

3. After storing the K_1, K_2 and ID_{RSU} into the TPD, controller send $< TPD >$ to the user.

4. The user finds the public key of RSU and controller as $A_1 \oplus K_1 \oplus n_u$, $A_1 \oplus K_2 \oplus n_u$ respectively and stores it for future communication. The user also generates its key pair (V_{pub}^i, V_{pri}^i) for secure communication. Figure 5.10 shows the graphical representation of the TPD and the values stored in the TPD.

5. Apart from all this, the controller shares its public key (Ctr_{pub}) and ID (ID_C) with all the connected RSUs and other controllers over the secure channel. This communication between the controller and RSU is also secured with ECC and SHA256.



(a) Initialization **(b)** Registration

Figure 5.11: (a) Initialization phase of LBSV, (b) Registration phase of LBSV

Registration Phase

The registration phase includes two components, i.e., vehicles and the controller. In this phase, the vehicle has to register with the controller for secure communication. Figure 5.11b depicts the registration process of the LBSV, which is as follows:

1. Vehicle V_i computes Rs, Rh, M_1, M_2 in the same sequence in which they are mentioned. Then the vehicle V_i sends $< ID_u^i, ID_v^i, M_1, M_2, n_u, T_1 >$ to the controller.

2. The controller first checks $\triangle T_1 \geq T_2 - T_1$, if it holds true then compute Rs^*, Rh^*, M_1^*, $Get(V_{pub}^i, Rs) = D(Ctr_{pri}(M_2))$, Kc^*. Then controller check if $(Rs^* = Rs \ \& \ M_1^* = M_1 \ \& \ Kc^* = Kc(stored))$ is true then the controller computes M_{info}, T_H, PBH. After computing M_{info}, T_H, PBH controller creates the block as $block = CreateBlock(T_H, T_3, PBH, BH)$ and sends it to the other controller nodes for the verification of the block. It uses a mPBFT *Algorithm* 14 to consensus on blocks. When the controller receives $2N/3$ commit messages, it adds the block into the blockchain and sends the $< confirmation, T_3 >$ (confirmation of the registration) back to the vehicle V_i.

3. Vehicle V_i receive the response and check if $\triangle T_2 \geq T_3 - T_1$ is true or not. If it is true, then update the TPD for communication; otherwise, discard the response and send the request again.

Consensus Algorithm: mPBFT

The registration phase and communication phase use the consensus algorithm to propose the vehicle registration phase block. The algorithm works as follows. The SDN controllers in the IoV network share messages to commit a block. Malicious nodes (SDN controller) might broadcast a tampered block into the network; thus, the block validated (committed) by at least 2N/3 nodes is considered a valid block. Here, N is the number of total nodes (considering that 1N/3 are faulty nodes). The algorithm uses four states, i.e., NEW-ROUND, VERIFICATION, COMMITMENT, and APPEND. The state's workflow goes from NEW-ROUND to COMMITTED. The consensus algorithm introduces the leader election process so that every miner node in the network gets equal chances to propose the block. Due to the new leader election process, the consensus delay is minimized compared with the other consensus algorithms, such as Proof-of-Work, Proof-of-Stake, and conventional PBFT. Every node is given a fair chance to append its block to the blockchain. The details of mPBFT are described in *Algorithm* 14.

1. The first state is NEW-ROUND. In this state, the block proposer is selected from all the miner nodes (controllers). The new transactions (Tr_{New}) are appended in the transaction pool (TP) as $TP.append = Tr_{New}$ for processing. A new block proposer is selected based on the following equation

$$miner = R(mod)N \tag{5.6}$$

   Here, R is the number of rounds in the current instance, and N is the total node (including faulty nodes). Round R is the one complete block creation process from the block proposing to be appended to the blockchain. This procedure of selecting the block proposer is a uniform distribution of chances to propose the block, i.e., every miner node (controller) gets an equal chance to create the block. Furthermore, this process is very lightweight, resulting in low consensus delay.

2. After the block proposer is selected, it creates a transaction block using the $CreateBlock()$ function and broadcasts it for verification. The state is now changed from NEW-ROUND to VERIFICATION.

3. In the verification state, all the other miner nodes, except the leader node, verify the block, i.e., all the transactions included in the block are verified. If the total number of verification is greater than $2N/3$, a message for commitment is broadcast into the network, and the state is changed from VERIFICATION to COMMITMENT.

4. The commitment is nothing but a confirmation of the block to append into the blockchain. If the total number of commitment messages is more than $2N/3$, the state is changed from COMMITMENT to APPEND.

5. In the append state, the proposed block is appended into the blockchain by all the miner (controller) nodes. The state is changed from APPEND to NEW-ROUND.

Algorithm 14: Consensus Algorithm: mPBFT

Input: New Transactions: $Tr_i, i = 1,2,\ldots,n$, Nodes: N
Output: Blocks
Function `newRound(`Tr_{New}, R, N`):`
    node.state = NEW-ROUND;
    $TP.append = Tr_{New}$;
    $select_miner(R,N)$; `/* Function to select the miner who creates the block */`
    $miner = R \mod N$; `/* R: round number.  N: Total nodes */`
    if $node$ == $miner$ **then**
        $block = CreateBlock(TP.getAllTransactions)$;
        $broadcast_block(block)$;
        node.state = VERIFICATION;
Function `verification(`$block$`):`
    $verify_block(block)$;
    if $(count(message.verification) \geq 2N/3)$ **then**
        $broadcast_message(message.commitment)$;
        node.state = COMMITMENT;
Function `commitement((`$message.commitment$`):`
    $commit_pool.append(message.commitment)$;
    if $(count(message.commitment) \geq 2N/3)$ **then**
        node.state = APPEND;
Function `append(`$block$`):`
    $blockchain.append = block$;
    node.state=NEW-ROUND;

Communication Phase

The SDN controller manages the public key and is granted to the vehicles through RSU, who want to communicate with other vehicles. The SDN controller verifies the identity of every vehicle requested before granting a public key. Every vehicle maintains and updates one list Ls_k, which is the list of the authenticated vehicle's public key and vehicle ID. The vehicle does not need to take a public key from the RSU and SDN controller at all times. Figure 5.12 depicts the key exchange and communication process in LBSV. The public key is not directly shared with the other vehicle. Instead, it is communicated via an SDN controller into the network. There are two reasons behind this process. The first is to receive information only from identified vehicles. Secondly, if any vehicle sends misinformation, it can be traced immediately to disqualify the network and be reported as a malicious vehicle. Legitimate vehicles can be protected from malicious vehicles using this key exchange method. Specifically, the key exchange and communication process is described below:

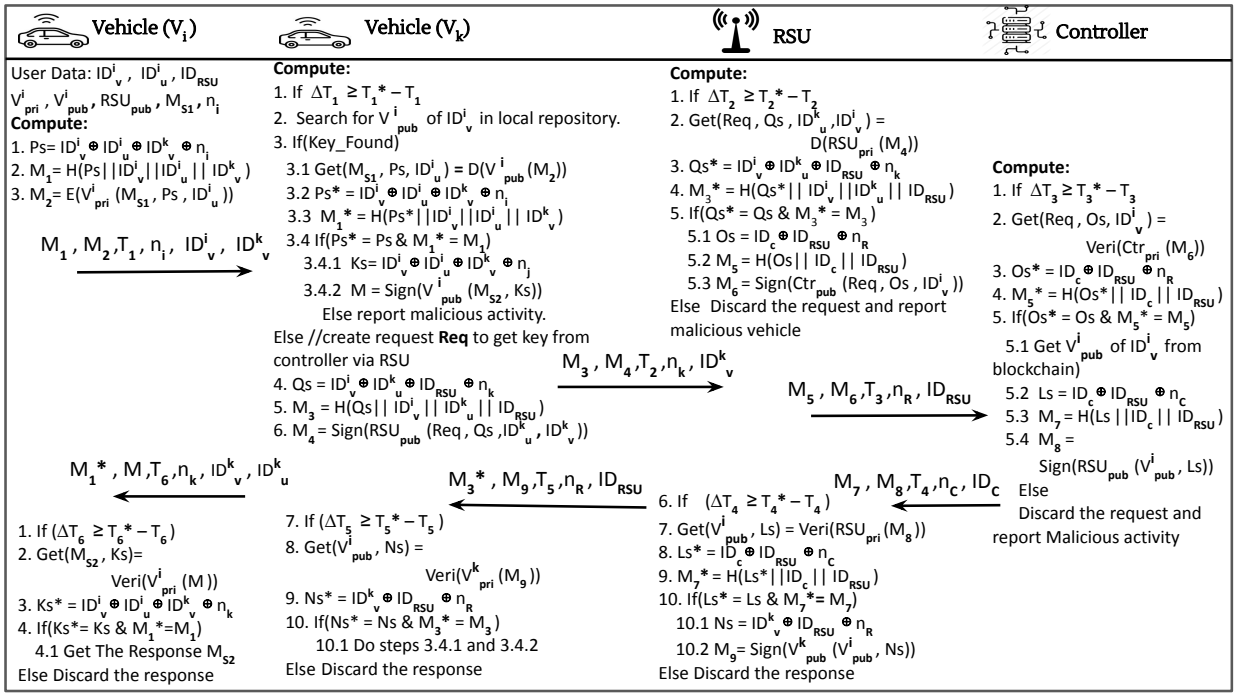


Figure 5.12: Communication phase of LBSV

1. Vehicle V_i generates the secret message M_{S1} and nonce n_i, and computes Ps, M_1, M_2. Vehicle V_i send $< M_1, M_2, T_1, n_i, ID^i_v, ID^k_v >$ to vehicle V_k.

2. Vehicle V_k receives the message from vehicle V_i. V_k first checks if $\triangle T_1 \geq T_1^* - T_1$ is true or not; if true, then proceed further. Otherwise, discard the response. If the condition holds true then vehicle V_k first look for the public key of vehicle V_i in local repository and do the following computation, $Get(M_{S1}, Ps, ID^i_u)$, Ps^*, M_1^* and then check if($Ps^* = Ps$ & $M_1^* = M_1$) is true or not, if true then do computation, Ks, M and response to the vehicle V_i. But if the key is not found at a local repository, the vehicle V_k sends a request to the controller via RSU to get the public key of vehicle V_i. The request is generated with Qs, M_3, M_4, and sends the $< M_3, M_4, T_2, n_k, ID^k_v >$ to RSU.

3. RSU receives the values and first it check if $\triangle T_2 \geq T_2^* - T_2$ is true or not. If it is true, then RSU does further computation else, it discards the request. If the above condition is true then RSU compute $Get(Req, Qs, ID^k_u, ID^i_v) = D(RSU_{pri}(M_4))$, Qs^*, M_3^*, then check if $(Qs^* = Qs$ & $M_3^* = M_3)$ is true then do further computation else report malicious activity. Then RSU does the following computation Os, M_5, M_6 and sends the $< M_5, M_6, T_3, n_R, ID_{RSU} >$ to the controller.

4. Controller receives the data and first checks if $\triangle T_3 \geq T_3^* - T_3$ is true, then do the following computation, $Get(Req, Os, ID^i_v) = D(Ctr_{pri}(M_6))$, Os^*, M_5^* and then check if $(Os^* = Os$ & $M_5^* = M_5)$ is true then go for further computation else discard the response and report malicious activity. If true then it compute $Get(V^i_{pub})$ of ID^i_v from blockchain, then Ls, M_7, M_8 and send the $< M_7, M_8, T_4, n_C, ID_C >$ to requesting RSU.

5. RSU receive the response and check if $\triangle T_4 \geq T_4^* - T_4$ is true then do the following computation, $Get(V^i_{pub}, Ls) = D(RSU_{pri}(M_8))$, Ls^*, M^*, then check if $(Ls^* = Ls$ & $M_7^* = M_7)$ is true then Ns, M_9 and RSU send the $< M_3^*, M_4, T_5, n_R, ID_{RSU} >$ to the vehicle V_j.

6. Vehicle V_k receive the data and check if $\triangle T_5 \geq T_5^* - T_5$ is true then vehicle compute the following, $Get(V_{pub}^i, Ns) = D(V_{pri}^k(M_9)), Ns^*$, then vehicle check if$(Ns^* = Ns \ \& \ M_3^* = M_3)$ is true then it compute Ks, M and send the $< M_1^*, M, T_6, n_j, ID_u^k, ID_v^k >$ to vehicle V_i.

7. Vehicle V_i receive the response and first check if $\triangle T_6 \geq T_6^* - T_6$ is true then do the following computation, $Get(M_{S2}, Ks) = D(V_{pri}^i(M), Ks^*$ and then check if $(Ks^* = Ks \ \& \ M_1^* = M_1)$ is true then get the response and act accordingly otherwise discard the response.

5.2.6 SmartCoin: A Novel Incentive Mechanism for Vehicles in Intelligent Transportation System Based on Consortium Blockchain

An ITS improves traffic safety and efficiency by enabling vehicles to generate and broadcast messages. However, trusting the received message's credibility in an untrusted environment or network is difficult. Additionally, there is no reward mechanism for the honest message generator. Hence, we proposed a novel incentive mechanism for vehicles called SmartCoin based on a consortium blockchain to improve safety and network efficiency and encourage more vehicles to participate in network improvisation. The proposed scheme aims to achieve social welfare, improved transportation, less traffic congestion, reduced road accidents, and a transportation network free from fraud messages. In the proposed scheme, vehicles give the rating to the message source vehicle based on the validity of the messages. Some SmartCoins are credited to the vehicle's account at the blockchain according to the message rating. Table 5.7 depicts different messages and their corresponding SmartCoins. The SmartCoins could be redeemed either at the fuel station, electric charge station for electric vehicles, or vehicle service station. Blockchain technology provides decentralization, immutability, and transparency to the ITS. The secure storage for the SmartCoin, traffic messages, and user information is achieved using blockchain. The proposed scheme encourages vehicles to participate and rewards them for their honesty. Vehicles rate the messages they receive, and based on that rating, some SmartCoins are credited to the vehicles' accounts in the blockchain. The scheme has two phases, i.e., the registration and communication phase. We have employed the consortium blockchain, where vehicles and RSU are the public nodes that generate, receive, and rate the messages but are not responsible for block mining. In contrast, the controller nodes make the private blockchain network, mine the blocks, and append them to the blockchain. We also explained the incentive mechanism of honest vehicles, which generates messages to improve network efficiency. We have defined the proposed Lightweight Consensus *Algorithm* (LCA) 15.

Message Types: In ITS, there are various types of traffic messages. Table 5.7 shows different types of messages and the corresponding value of SmartCoin (STC), which is to be credited into the vehicle account if messages are generated by the vehicles. The messages are arranged in priority order. $M1$ has the highest priority, so the SmartCoins are, and $M7$ has the lowest priority. The priority is assigned based on the emergency to take action. The Accident is a life-lost event, so it has the highest priority. Similarly, priority is assigned to other types of messages. The STC is the representation unit for SmartCoins like BTC in Bitcoin cryptocurrencies.

Registration Phase

For communication with other vehicles and RSUs, the vehicle must first register at the controller node. The communication channel is secure in the registration phase. The registration process is as follows:

Table 5.7: Types of messages and the corresponding SmartCoins

Symbol	Meaning	Credited SmartCoin (STC) into Vehicle/User Account
$M1$	Accident occurred	100 STC
$M2$	Accident area	50 STC
$M3$	School zone	30 STC
$M4$	Malicious information	20 STC
$M5$	Traffic jam	15 STC
$M6$	Fuel station	10 STC
$M7$	Road intersection	5 STC

1. First, vehicle V_i computes $A_i = (H(Y_i)||M_{req}||V^i_{pub})$ and $B_i = A_i \oplus ID^i_v$, and send the $< ID^i_v, B_i, T_1 >$ to the controller node.

2. The controller node first checks the $\Delta T_1 \geq T_2 - T_1$, here T_2 is the current timestamp of controller node. Then, if the condition is satisfied, the request is processed further. It computes $C_i = B_i \oplus ID^i_v$ and extracts the $H(Y_i)$. The controller node connects to the factory database, gets the Y_i corresponding to ID^i_v, and checks $H(Y_i)' \equiv H(Y_i)$.

3. If both $H(Y_i)$ matched, it compute, $D_i = H(C_i) \oplus (Ctr_{pub}||H(Y_i))$ and $E_i = H(C_i) \oplus (RSU_{pub}||H(Y_i))$. Controller node sends $< D_i, E_i, T_3 >$ to request vehicle V_i over the secure channel. After that, it creates the transaction as $Tr_1 =< (ID^i_v \oplus V^i_{pub}) >$. The same process followed for other vehicles as well. Hence, the controller has many transactions $< Tr_1, Tr_2, ..., Tr_n >$. The controller now creates the block after broadcasting the transactions to other controller nodes in the network using a digital signature. The other nodes verify this transaction and keep it in their transaction pool.

4. To create the block, controller node compute, $Mrt()$= $H(H(Tr_1), H(Tr_2), ..., H(Tr_n), BH, PBH$ and create the block $Block = createBlock(B_{ID}, Mrt(), PBH, BH, T_2, < H(Tr_1), H(Tr_2), ..., H(Tr_n) >)$. The controller node sends B_i to other nodes for verification. After verification, $Block$ is appended to the blockchain ledger by all the controller nodes so that the copy of the blockchain is identical to all the nodes.

5. The controller nodes also share the $< ID^i_v, V^i_{pub}$ with all connected RSUs using signature and verification, so that the vehicles get the public key of other vehicles from RSUs, i.e.,$\sigma = SIGN(Ctr_{pri}, (ID^i_v, V^i_{pub}))$, $VERIFY(Ctr_{pub}, (ID^i_v, V^i_{pub}), \sigma)$.

6. The requesting vehicle receives $< D_i, E_i, T_3 >$ from controller node. Then it checks $\Delta T_2 \geq T_4 - T_3$. Then, also check $H(A_i) \equiv H(C_i)$ and $H(Y_i) \equiv H(Y_i)$, and if both found true, then it extracts Ctr_{pub} and RSU_{pub} and store them into TPD.

Communication Phase and Credit of SmartCoin into Vehicle Account

As explained in the registration phase, the vehicle's public key, RSU, and controller node are shared. The wireless access in vehicular environments (WAVE) protocol is being used for communication in the ITS. Figure 5.13 shows the communication of V2V, V2R and R2N. While running on the road, the vehicle observes the events mentioned in Table 5.7 and broadcasts them into the network, either V2V or V2R. The communication channel in this phase is open. The communication can be defined as follows:

1. Vehicle V^i observe the event, e.g., M_i. After that, the vehicle sends this message to either another vehicle or RSU, whichever is in the communication range. Vehicle computes $\sigma_{M_k} = SIGN(V^i_{pri}, M_k)$ for $k = 1, 2, ..., 7$ and sends to either RSU or another vehicle. In any case, i.e., RSU or vehicle received a message is verified using the vehicle's public key as $VERIFY(V^i_{pub}, M_k, \sigma_{M_k})$. If the output is true, it is checked whether the received message is genuine or fake. If it is genuine, a one (1) rating is given; otherwise, a zero (0) rating is given.

2. The response to the received message is as follows. If genuine message is found then $\sigma_{(M_k||1)} = SIGN(V^j_{pri}, (M_k||1))$, and if fake message is found then $\sigma_{(M_k||0)} = SIGN(V^j_{pri}, (M1||0))$. the same response is also sent to the RSU (response signed by RSUs public key RSU_{pub}) so that the genuine rating can be recorded.

3. If the positive rating is more than 50%, then the SmartCoin corresponding to the message (mentioned in table 5.7) is credited into the vehicle's account and stored in the blockchain. For example in case of message type $M1$, which we have taken, the 100 STC as:
SmartCoin:1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA (The address of vehicle account)
amount=100
label=warning message
message=accident occurred
A transaction is created and added to the UTXO (unspent transaction list) of the vehicle account so that these coins can be redeemed at the fuel station or service station. The UTXO of the vehicle account looks like

```
[
    {
        "txid" : "9ca8f969bd3ef......c309acbae2c14ae3",
        "SmartCoin" : "1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA",
        "scriptPubKey" : "76a91407bdb51....c9c13d1fd288ac",
        "Script"    :   "188192f....481f9fa77d116f39222a",
        "amount" : 100,
        "confirmations" : 200
    }
]
```

Lightweight Consensus Algorithm

In SmartCoin, the controller node makes the private blockchain, and the vehicles and RSU work as the public network to help authenticate the generated information. Thus, it is called consortium blockchain. The controller nodes mine the block, and the consensus algorithm is used to propose the block into the blockchain, which contains the network traffic messages and SmartCoins. Though the controller node network is a private blockchain network with only trusted controller nodes, there is no need to apply any competition amongst the nodes for block proposals like PoW. Therefore, we apply the Round Robin scheme, where every controller node proposes a block in the given time quantum δT. All the nodes are arranged in order, and every node is allowed to propose a block when δT of the previous block has elapsed. The *Algorithm* 15 works as follows for block creation and verification.

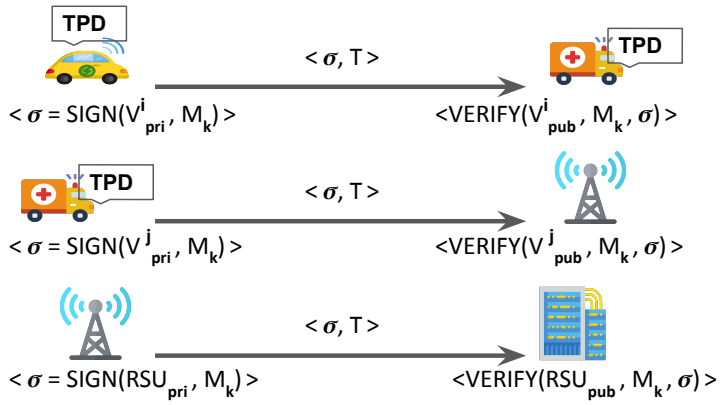


Figure 5.13: Communication between Vehicle, RSU, and Controller Node

1. The public nodes vehicles and RSUs are the traffic messages originator that generates the message based on the type of condition (See Table 5.7) and put them in the *transaction_pool* at the controller node.

2. The controller node assembles all the transactions from the *transaction_pool* and verifies them with the help of public node participants, i.e., vehicle. The recipient controller node then broadcasts these verified transactions to other controller nodes. After that, the selection block proposer calls the function *createHashTree* to calculate the *Mrt* of these transactions.

3. Next, the block proposer controller node also calculates the *PBH* using the *getHash* function along with the current timestamp T.

4. Finally, the block proposer controller node gathers all the calculated parameters, which are *Mrt*, *PBH*, T and *nonce* and generates the block. A *createBlock*() function is used to generate the function. After generating the block, the controller node broadcasts it to the network for verification. The block generator signs a block.

5. Other than the proposer node, other nodes are responsible for verifying the newly generated block, which is first verified using the generator's public key. To validate that the generated node contains legitimate information/transactions, first, they calculate the *Mrt* from the *transaction_pool* and check whether the calculated block matches the received block. If the match is correct, the *block_status* is set as *valid*; otherwise set as *invalid*. The *node.status* is also set as bad, so the node is not considered for the next block proposer.

6. The block proposer controller node confirms the new block if it receives at least $3N/4$ *valid* response where N is the total node. This means that if 75% of the node verifies, the block is considered the valid node.

5.2.7 Implementation Environment and Performance Analysis

This section explains the proposed scheme's implementation environment and performance analysis compared with state-of-the-art schemes. The performance analysis includes computation cost, energy consumption, storage cost, communication cost, and consensus delay. We have implemented the scheme using

Algorithm 15: Lightweight Consensus Algorithm

Input: Transaction: $Tr_1, Tr_2, ...Tr_n$
Output: block
$TP.append = Tr_i$; `/* append all the new transactions in the transaction pool */`
Function `blockGenerate(`Tr_{New}`):`
  if *node* == *proposer* **then**
    $Mrt = createHashTree(TP.getAllTransactions)$;
    $T = Timestamp$;
    $nonce = random_number$;
    $PBH = getHash(previous_block)$;
    $BH = getHash(Mrt, T, PBH, nonce)$;
    $block = createBlock(Mrt, T, nonce, PBH, BH)$;
    $sign_block(block)$;
Function `blockVerification(`*block, nonce*`):`
  if *node != proposer* **then**
    $receive_block(block, nonce)$;
    $Mrt = createHashTree(TP.getAllTransactions)$;
    $block* = createBlock(Mrt, T, nonce, PBH, BH)$;
    if $block == block*$ **then**
      $block_status(valid)$;
    else
      $block_status(not_valid)$;
      $node.status = bad$;
  if $(count(block_status == valid) \geq 3N/4)$ **then**
    $blockchain.append(block)$;

Raspberry Pi devices and produced results that are promising and convincing. Below, we have explained the analysis in detail.

Implementation Environment

The performance assessment for LBSV and SmartCoin is conducted using the Raspberry Pi 4 and Windows OS-based Desktop. The hardware configuration of Raspberry Pi is as follows, Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz processor, 2 GB memory, standard 40-pin GPIO header, and Linux-based Noobs v3.3.1 operating system. The hardware configuration of the Windows desktop system is Intel i5 processor 9th Generation, 8 GB RAM, 512 GB SSD secondary storage, Windows 11 operating system, 4 GB Graphics card of NVIDIA GeForce GTX 1650. Java programming is used to script blockchain consensus algorithms and cryptographic functions. Netbeans IDE 11 (64-bit) and Java 8 (Oracle Java SE, 64-bit) frameworks are used to perform the experiment. We have considered Raspberry Pi, 2GB as the lightweight device (i.e., OBU and RSU) and Desktop System as the controller node.

Table 5.8: Individual computation time of different cryptographic functions in the testbed environment

Functions	On Desktop	On Raspberry Pi 2GB
SHA256 Hash function: $T_{H()}$	0.0008	0.045
ECC Crypto-system: T_{ECC}	0.053	3.9
ECDSA signature: T_{sign}	0.071	6.13
ECDSA verification: T_{veri}	0.065	5.63
point multiplication: $T_{mul(.)}$	0.117	11.0
AES Crypto-system 192-bit: T_{AES}	0.125	12.71
Modulo operation: $T_{mod(.)}$	0.067	5.0

Computational Cost and Energy Consumption

Computational time is the time taken by the simulator for the execution of the whole framework. The measuring unit is milliseconds (MS) and represented as T_{ct}. We illustrate the computational cost (also known as execution time) based on the operational cost for authentication, verification, and communication. The different cryptographic functions used such as $\oplus$: bitwise XOR, T_{AES}: AES encryption/decryption 192-bit, T_{ECC}: ECDSA encryption/decryption, $T_{H()}$: SHA 256 hash function, T_{sign}: ECDSA signature, T_{veri}: ECDSA verification, $T_{mul(.)}$: point multiplication, $T_{mod(.)}$: Mod operation (Assuming that XOR and concatenation operations take negligible time compared to other operations). Table 5.8 shows the execution time of different cryptographic functions. Table 5.9 shows the comparative computation cost of SmartCoin and LBSV with state-of-the-art schemes. The energy consumption is the energy consumed by the system to perform the operations of the scheme. In the execution process, the system requires some energy to carry out all the phases of the scheme. The energy consumption is calculated as $E_{rg} = T_{ct} * C_P$, where C_P is a CPU maximum power (10.88 W for a laptop and 2.4W for Raspberry Pi 4) and T_{ct} is a computational time. It is measured in Joules We have calculated the computational cost in the previous section. Here, we calculate the energy consumption. We calculated the energy consumption on two platforms, i.e., Desktop and Raspberry Pi. Table 5.9 shows the comparative analysis of LBSV and SmartCoin energy consumption with state-of-the-art schemes. It is very clear from the table that energy consumption is reduced in LBSV and SmartCoin compared to state-of-the-art schemes.

Table 5.9: Computation cost (CC) and energy consumption (EC) of LBSV and SmartCoin compared with state-of-the-art schemes on Desktop and Raspberry Pi (RP)

Scheme	Computation Cost Equation	CC on Desktop	CC on RP4	EC on Desktop	EC on RP4
[156]	$10T_{H(.)} + 4T_{sign} + 4T_{veri}$	0.552	47.44	6.01	113.85
[157]	$8T_{H(.)} + 3T_{ECC} + 3T_{sign} + 3T_{veri}$	0.572	47.80	6.24	114.72
[110]	$5S_{H(.)} + 3T_{sign} + 3T_{veri}$	0.548	47.35	5.96	113.64
[158]	$5T_{H(.)} + 2T_{sign} + 2T_{veri}$	0.276	23.74	3.01	57.00
[120]	$14T_{H(.)} + 2T_{sign} + 2T_{veri} + 1T_{mul(.)}$	0.400	35.15	4.35	84.36
LBSV	$15T_{H(.)} + 2T_{ECC} + 2T_{sign} + 2T_{veri}$	0.390	31.99	4.24	76.56
SmartCoin	$6T_{H(.)} + 4T_{sign} + 4T_{veri}$	0.410	35.05	4.46	84.12

Storage Cost and Communication Cost

Storage cost refers to the storage required for storing the values such as secret key, hash value, and other parameters in all phases. The measuring unit for storage cost is bytes and represented as S_{sc}. Generally, a Key S_k requires 24B (B=Bytes), ECC encryption/decryption S_{ECC} takes 48B, AES encryption/decryption S_{AES} takes 24B, a hash function (SHA-256 bit) $S_{H(.)}$ takes 32B, and ECDSA based signature S_{sign} and verification S_{veri} requires 10B. Table 5.9 and Figure 5.14a represent the comparative storage cost of the LBSV and Smartcoin with state-of-the-art schemes. Communication cost refers to the storage required for storing the

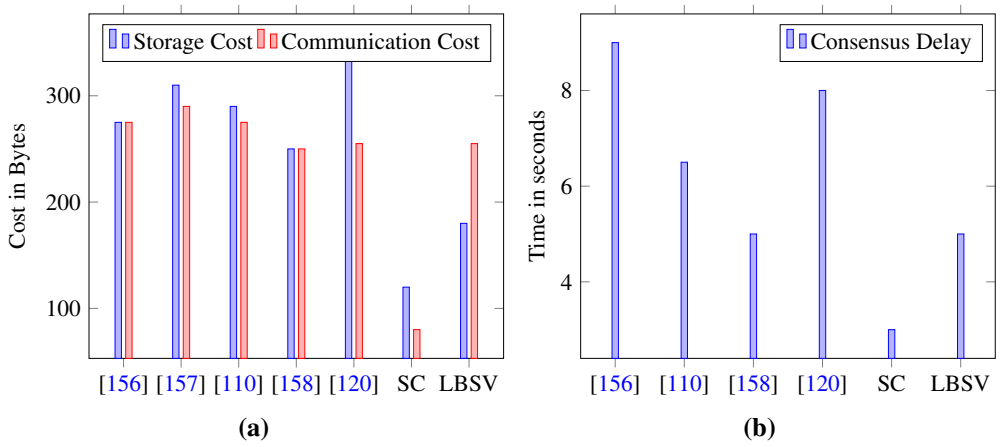


Figure 5.14: (a) Storage and communication cost of SmartCoin (SC) and LBSV with state-of-the-art scheme (b) Consensus delay of SmartCoin (SC) and LBSV with state-of-the-art scheme

Table 5.10: Comparative storage and communication cost of LBSV and SmartCoin with state-of-the-art schemes

Scheme	Storage Cost	Cost in Bytes	Communication Cost	Cost in Bytes
[156]	$8S_{H(.)}+2S_{sign}$	275	$8S_{H(.)}+2S_{sign}$	275
[157]	$6S_{H(.)}+3S_{ECC}+2S_{sign}$	310	$5S_{H(.)}+3S_{ECC}+2S_{k}$	290
[110]	$8S_{H(.)}+4S_{sign}$	290	$7S_{H(.)}+5S_{sign}$	275
[158]	$6S_{H(.)}+6S_{sign}$	250	$6S_{H(.)}+6S_{sign}$	250
[120]	$10S_{H(.)}+1S_{sign}+S_{mul(.)}$	350	$7S_{H(.)}+1S_{sign}+S_{mul(.)}$	255
LBSV	$3S_{H(.)}+2S_{Sign}$	116	$6S_{H(.)}+2S_{ECC}$	240
SmartCoin	$3S_{H(.)}+2S_{sign}+1S_{ECC}$	164	$3S_{H(.)}+3S_{sign}$	126

communicating variables such as hash code, ciphertext, and other parameters. The measuring unit is bytes and represented as C_{cc}. The operations and the variables that have been used in communication between candidates and controllers are a hash function, ECDSA based digital signature. The storage requirement for different cryptographic operations is mentioned in the previous section. Table 5.10 and Figure 5.14a represent the comparative storage cost of the LBSV and SmartCoin with state-of-the-art schemes.

Consensus Delay

The LBSV used the mPBFT, and SmartCoin used the LCA consensus algorithm for block creation and verification. A fair leader selection is very important because if only fewer nodes are selected repeatedly, it will attract the attackers and behave like a centralized system. For a fully decentralized system, all the nodes should have an equal chance to propose a block. Therefore, we employ the simple yet effective leader selection process to elect a leader to propose the block. Due to this mechanism, the consensus delay of the LBSV and SmartCoin is minimized. Furthermore, in SmartCoin, the message communication overhead in LCA is less than that of LBSV. Therefore, the SmartCoin shows the lowest consensus delay. The state-of-the-art schemes also use different consensus algorithms for block verification and validation such as delegated proof-of-stake (DPoS) used in [158], in [110], PoW and PBFT are used. In [120], PBFT is used for message verification under the hyperledger fabric framework. Among all the consensus algorithms used by all these schemes, the mPBFT and LCA perform block creation, verification, and validation in the

least time, which is much less than the others. Figure 5.14b shows a comparative analysis of LBSV and SmartCoin with state-of-the-art schemes.

SDN Computational Efficiency

We also calculated the computational efficiency of the SDN controller. An SDN controller's computational efficiency can be defined as the time taken to respond to a communication request and blockchain creation process. Figure 5.15 represents the computational efficiency of our proposed scheme. It can be evident from the figure that the computational cost is meager in the low-dense scenario because of the small chaining of the blockchain. Due to fewer communicating parties, verifying the blocks and the creation of the blockchain is easy. In the higher denser scenario, the computation cost increased by a small margin. This is a small price to pay for the benefits we are getting from the proposed protocol.

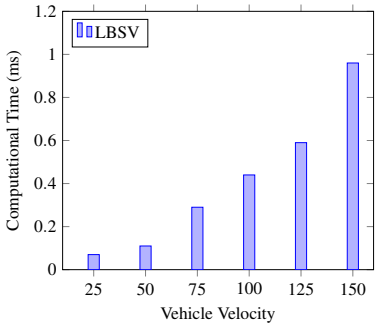


Figure 5.15: Computational efficiency of SDN

5.2.8 Security Analysis

In this section, we explain the security analysis of LBSV and SmartCoin based on the cryptographic and blockchain aspects. Theorems 1, 2, and 3 claim the cryptographic aspect, and Lemmas 11, 12, and 13 claim the blockchain aspect. Before proving the blockchain aspect, we have also provided the attacker scenario for more clear understanding.

Attacker Scenario

The attack scenario is extended compared to Nakamoto's consideration. Here, we add a finite-time attack model, i.e., a time gap exists for the possible attack. Other than the honest miners, there are also some adversaries or attackers. The honest miners tend to the honest longest chain, and the attacker tends to the fraudulent chain. To launch the double-spending attack, the attacker targets the transaction to pay for goods and services (such as in SmartCoin, the services are a service station, fuel station, and an electric vehicle charging station). A target transaction is a transaction made by an attacker to transfer cryptocurrency from the attacker to the victim. Let, at time t, the last block of the honest chain is generated. At the time $t+i$, the attacker broadcasts the fake transaction in the network to be included in the next block, puts it into the honest chain, and simultaneously creates the fork in the honest chain, which derives from the last block secretly. This secret fork is referred to as a fraudulent chain containing the attacker's fake transactions. A block confirmation time is when the victim is waiting before providing any service (which should be minimal in real-time applications like IoV). Let N_{BC} be the number of blocks chosen to be a block confirmation time.

The fraudulent chain (a fork in the honest chain) is made public once it is longer than the honest chain after the block confirmation time. The SmartCoin withstands double-spending attacks, fork attacks and fault tolerance, and we proved this in the following lemma considering the above attack scenario.

Theorem 1. *The SmartCoin supports the authenticity of the data shared between TPD and the controller node.*

Proof. The digital signature is secured under random message attack if for the message (transaction of SmartCoin) space M and all adversaries $\mathbb{A}$ the success possibility of $\mathbb{A}$ is negligible. Let, $M1, M2, ...Mn$ are the uniformly chosen messages at random from message space M and $KEYGEN(k)$ generates the private-public keypair (V^i_{pub}, V^i_{pri}). Let the adversary compute the digital signature of above mentioned chosen messages as $\sigma_{M1} \leftarrow SIGN(V^i_{pri}, M1), \ldots \sigma_{Mn} \leftarrow SIGN(V^i_{pri}, Mn)$. The adversary $\mathbb{A}$ is given V^i_{pub} and $(Mi, \sigma_{Mi})^n_{i=1}$ and output (M, σ_M). Adversary $\mathbb{A}$ is succeeded if $VERIFY(V^i_{pub}, X, \sigma_X) = True$ and $X \notin (M1, M2, ..., Mn)$. The scheme is strongly protected under a random message attack if the following condition is satisfied. $VERIFY(V^i_{pub}, X, \sigma_X) = True$ and $(X, \sigma_X) \notin \{(M1, \sigma_{M1}), (M2, \sigma_{M2}), ..., (Mn, \sigma_{Mn})\}$. In SmartCoin, an adversary $\mathbb{A}$ could observe the communication channel. However, messages (M) and transactions (Tr) from the TPD are signed $\sigma_{Tr/M} \leftarrow SIGN(V^i_{pri}, tr/M)$ and verified $VERIFY(V^i_{pub}, Tr/M, \sigma_{Tr/M})$ by each receiver before further process. An adversary cannot calculate the hash of the transaction. Even if the adversary hijacks the channel and gets the signed messages, the corresponding verification cannot be done. At the same time, blockchain guarantees the authenticity of stored information. □

Theorem 2. *The SmartCoin supports data integrity of the network traffic messages, user information, and transactions.*

Proof. SmartCoin employs the signature and verification algorithm. Suppose an adversary is able to know the transactions and try to forge them. In that case, the adversary again has to sign the transaction, which requires the private key of the vehicle or controller node whose transaction is to be modified by an adversary, which is not possible. Hence, the scheme provides the integrity of the transactions—furthermore, the blockchain records the information in the form of a block, which is immutable and tamper-proof. Therefore, the integrity of the information contained in the block can be assured. The attack is possible only if the adversary successfully acquires 51% of the legitimate nodes. □

Theorem 3. *The SmartCoin supports non-repudiation of the network traffic messages, user information, and transactions.*

Proof. With the help of proof of origin (PoO) and mandatory proof of receipt (PoR) [159], it can be justified that SmartCoin supports non-repudiation of network traffic messages in the ITS network. A mutual non-repudiation is achieved by encompassing the mandatory PoO and PoR both. The PoO must associate the originator's identity with exchanged information to not deny it after generating the message. Similarly, the PoR is also associated with the recipient's identity. The association in both must be undeniable and unforgeable. We have achieved both properties using the ECDSA-based digital signature. The proof can be described as follows.

$$PoO = Sign(V^i_{pri}[\mathbb{O}_V, V, RSU, M_i], T_1), \forall i = 1,2,3,4,5,6,7 \tag{5.7}$$

$$PoR = Sign(RSU_{pri}[\mathbb{O}_{RSU}, RSU, V, H(PoO)], T_2) \tag{5.8}$$

where $\mathbb{O}_V, \mathbb{O}_{RSU}$ indicates that this structure is represented as PoO and PoR, respectively. V and RSU are the originator and recipient (i.e., Vehicle and RSU), respectively. T_1, T_2 are the timestamps and M_i is the type of message where i varies from 1 to 7. In the PoO and PoR, the identity of vehicles and RSUs are associated, which is explained in the registration and communication phase of the SmartCoin scheme. Thus, the above two equations clearly show that the proposed scheme supports non-repudiation. It is very important to show it because it helps identify the malicious vehicles that try to broadcast fake messages. Additionally, the immutability of the signature assures the non-repudiation of the transaction. The controller node puts all the transactions in the block and publishes them in the blockchain. The vehicle can obtain the SmartCoin corresponding to its unique ID and private key. Once the information is published into the blockchain, it can not be modified. Hence, the SmartCoin withstands non-repudiation. □

Lemma 11. *The SmartCoin protocol is protected from double-spending attacks. A double-spending attack is possible if there exists a double-spending attack time $t_{dsa} \in [0, \infty]$ such that, in this duration, the fraudulent chain length is longer than that of honest chain length.*

Proof. It is termed double spending when someone spends the same amount of money twice. Each node in the network verifies every transaction in the digital cryptocurrency, allowing for multiple copies of the transaction to be produced. The transaction is then re-broadcast into the network prior to final transaction confirmation. Double-spending attack is a serious issue for blockchain systems such as Bitcoin, Bitcoin-NG, and others. Double-spending can be understood from the process described below. (Note: UTXO is the unspent output transaction with respect to the cryptocurrency). Let M be a victim, and Λ be an adversary, and the double-spend attack has happened when N_{BC} have validated within a period of time of t_{dsa}.

Let, at time t = 0, a M receives a transaction Tr from Λ. The Tr is an unspent output transaction issued from a UTXO. The fair Miners mine the block transparently, and Λ mine secretly, it includes Tr; however, Λ does not include Tr, but another transaction (fake transaction) Tr' conflicting with Tr. Also, when N_{BC} have mined, M sends goods to Λ. The Λ has mined the blockchain greater than the legal one, it sends his blockchain to the network. Thus, the legitimate transaction Tr is removed from the blockchain.

The double-spending attack is possible if t_{dsa} is in $[0, \infty]$ range. However, the block confirmation time is very low in SmartCoin, and the consensus delay is also very low (in milliseconds). Hence, the t_{dsa} is not possible. To produce a block, just the leader node is permitted. Since this eliminates the chance of rebroadcasting the same transaction, SmartCoin eliminates the potential of rebroadcasting the same transaction in the network. □

Lemma 12. *The SmartCoin protocol prevents fork generation. A fork in the blockchain is possible if $t_{fork} \in [0, \infty]$ such that, in this duration, the fraudulent chain length is longer than that of honest chain length.*

Proof. The difference between the number of nodes mined by the malicious node and the fair node to confirm a transaction should be < 0, i.e., $t_{fork} < 0$. Additionally, to lower the vulnerability probability of the blockchain network, the threshold $\varepsilon \in (0, 1]$ is the effective, robust level. This can be defined as follows:

$$\theta^*(\varepsilon) = min\{\theta : P(Q > \theta) \leq \varepsilon\} \tag{5.9}$$

where P is the probability of a malicious node to mine the block before the fair node. The $\theta^*(\varepsilon)$ can be calculated as

$$\theta^* = -\log(\varepsilon)/\log(\lambda/\Lambda) \tag{5.10}$$

The high value of θ^* has the efficiency to counterattack the intentional forks. The computational power of the fair node needs to improve to strengthen the robustness of the blockchain network. The fair node's effective computational power is the minimum to complete the mining in the due time interval. In the case of SmartCoin, the computation power of the block proposer to derive a new block in the time interval is high from the malicious node. Hence, an adversary cannot create a fork in the SmartCoin. Bitcoin, Bitcoin-NG, ByzCoin, Algorand, and OmniLedger all have problems with fork generation, but in SmartCoin, nodes create blocks at a fixed interval. The next node in the network starts creating the blocks when the interval is over. SmartCoin has removed the issue of needless fork generation because there is no chance of two miner nodes coming up with the same block at the same time. There is no parallel block generation in SmartCoin. Thus, there is no possible way for multiple nodes to submit a block simultaneously, thus no fork creation. □

Lemma 13. *The SmartCoin protocol is robust and fault-tolerant.*

Proof. The SmartCoin is the blockchain-based protocol and employs the consensus algorithm 15 to propose the block for the new set of transactions. The robustness and fault tolerance are the system's properties that enable it to function correctly even when some of the nodes behave byzantine. The high fault-tolerant system's performance continues to be consistent even though some nodes of the system failed to perform their task. The SmartCoin is robust and fault-tolerant in terms of topology as well as node failure or node compromising. As illustrated in Figure 5.16, if any node is not functioning correctly or becomes a dead node, the network still not be affected. Thus, the failure of any node does not affect the SmartCoin's working. Hence, SmartCoin is fault-tolerant. The proposer miner node broadcasts the block in the network for verification. All the nodes in the network verify the blocks and set their status as valid. The leader node adds the blocks only after receiving 3N/4 valid responses, i.e., if there are N nodes, and some of the nodes are behaving byzantine, then the system will not fail. However, if the failed nodes are more than the $3N/4$, then it will degrade the system's performance. □

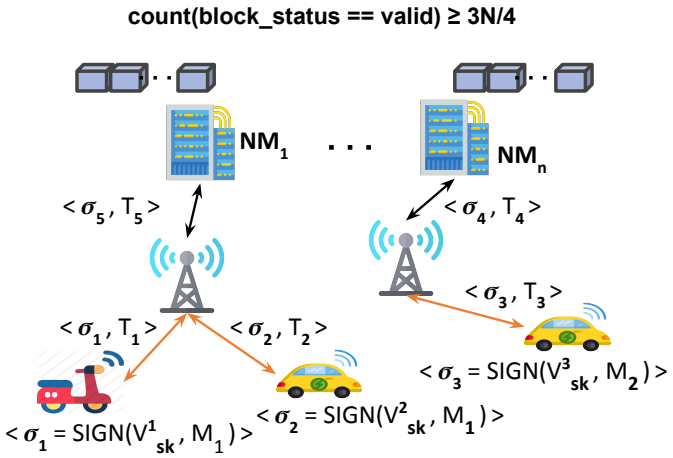


Figure 5.16: Fault tolerance in SmartCoin

A. Identification Theft

Identity theft is the intentional use of legitimate users' identity, usually as a way to achieve an advantage. In the LBSV user first register vehicle V_i to controller as, $A_1 = H(ID_u^i || ID_v^i)$ and $A_2 = A_1 \oplus n_u$, then controllers do $Get(A_1) = A_2 \oplus n_u$. Then it calculate $K1$,$K2$ and Kc as $A_1 \oplus RSU_{pub} \oplus n_u$, $A_1 \oplus Ctr_{pub} \oplus n_u$ and $n_u \oplus A_1$ respectively. At the time of registration, the value of Kc received from the vehicle and stored at the controller is matched by the controller; if it is found unmatched, then the malicious activity is reported to all the RSUs and controllers. Hence, communication with a fake identity or duplicate identity is impossible, i.e., only legitimate and identified vehicles can communicate with other vehicles, RSUs, and SDN controllers.

B. Message Integrity

This attack happens when someone can modify the message's content in transit, i.e., integrity violation. In the LBSV, the interaction between vehicles (TPD), RSUs and SDN controller is secured with the hash (SHA256) and ECC-based cryptographic functions. Vehicle V_i create hash $M_1 = H(Ps||ID_v^i||ID_u^i||ID_v^j)$, and do encryption of message $M_2 = E(V_{pri}^i(M_{S1}, Ps, ID_u^i))$. Then the vehicle V_i sends $< M_1, M_2, T_1, n_i, ID_v^i, ID_v^j >$ to another vehicle. Vehicle V_j checked the received hash with calculated hash as $M_1^* = M_1$ if found unmatched, then reported the malicious activity to the RSUs and controllers about V_i. The message is encrypted with the vehicle's private key, so attackers cannot modify the transit content.

C. Sybil Attack

In the cyber world, the Sybil attack is a new threat, especially in the IoT and IoV, where a fake entity tries to take control of the network by flooding fake nodes. In LBSV, each vehicle ID along with its public key is stored at controllers as $M_{info} = V_{pub}^i || ID_u^i || ID_v^i$, and $T_H = H(M_{info})$. Then, after it creates a *block*. Hence, the device can have only one identity, and each vehicle can have only one key pair at a given time. Before creating the block at the controller, we did one more check as $Kc^*(received) = Kc(stored)$, and if found unmatched, then a malicious vehicle was reported. The entity's key pair (i.e., vehicle, RSU, SDN controller) secures the entire communication. Moreover, the participating vehicles are first authenticated by the SDN controller; only then are they able to communicate with other components of the network. Thus, an attacker cannot use fake identities.

D. Message Replay Attack

When an attacker sends the same message the legitimate user already sent, this attack is called a replay attack. This attack is difficult to detect because the message is not modified; therefore, the receiver accepts it as a genuine message and takes actions desired by the attacker. This attack is impossible for two reasons: first, We have used the blockchain concept, which eliminates the double-spending problem, and second, we use the timestamp and a threshold value for the message. If the attacker sends duplicate messages, the receiver discards them as the condition $\triangle T >= T_i - T_j$ will not hold. Hence, a replay attack is not possible with our scheme. In LBSV, timestamp checking is done between the communication of every component and all the phases of the scheme.

Table 5.11: Comparative security analysis of LBSV and SmartCoin with state-of-the-art schemes based on different attacks

Attacks	[156]	[157]	[110]	[158]	[120]	LBSV	SmartCoin
Identification:	✓	✓	✓	✓	✓	✓	✓
Message integrity:	✓	✓	✓	✓	✓	✓	✓
Sybil attack:	✓	×	✓	✓	×	✓	✓
Message reply attack:	✓	✓	✓	✓	×	✓	✓
Botnet attack:	×	✓	✓	✓	×	✓	✓
Man-in-Middle attack:	✓	×	✓	×	✓	✓	✓
RF Interference/Jamming:	✓	✓	×	×	×	✓	✓
Fake Node injection:	✓	✓	✓	×	✓	✓	✓
Tampering:	✓	×	✓	✓	×	✓	✓

E. Botnet Attack

It is the most dangerous attack in IoV. A botnet works under the command and control of the bot master. It is a network of compromised machines (or bots) that run malicious software. Botnets have a broad spectrum of wicked purposes, including email spamming, DoS/DDoS attacks, password guessing, key-logging, and cryptocurrency mining. The LBSV is protected from the botnet attack because the vehicle becomes part of the IoV only after the proper authentication, i.e., the controller issued a TPD to the vehicle after the initialization phase, which consists of RSU_{pub}, Ctr_{pub}, ID_C and ID_{RSU}. In addition to this vehicle, V_i generates a key pair V^i_{pri}, V^i_{pub} and stores it into the TPD. Therefore, a botnet attack is not possible.

F. Man-in-The-Middle (MiTM) Attack

It is assumed that the ongoing communication between two or more entities is compromised and changed by unauthorized or unregistered entities, then we say that a man-in-the-middle attack has occurred. In the LBSV, V_i sends information to all nearby connected vehicles and RSUs via the open channel (insecure channel), which consists of $< M_1, M_2, T_1, n_i, ID^i_v, ID^j_v >$ during message transmission. If an unauthorized vehicle V_m wants to know S_M, then it should exactly know M_2, but it is encrypted using the private key of vehicle V_i, so V_m cannot see the content of the message.

5.3 BLOCKCHAIN-BASED SECURED SMART GRID SYSTEM

Distributed renewable energy resources, such as solar panels, batteries, electric vehicles, adaptable loads, and other modern technologies, are increasingly being incorporated into the smart grid as an energy distribution system. For instance, the current global market for rooftop photovoltaic (PV) panels is projected to be worth about $33 billion, with an expected growth of more than 3.7 GW by 2025 due to societal adoption [160]. Utilizing these additional energy resources is essential for controlling energy demand and significantly integrating renewable energy into the smart grid. The smart grid merges internet and communication technologies (ICTs) with power networks, enhancing energy generation, transmission, and distribution to be more efficient, cost-effective, and environmentally friendly [161]. The smart grid represents a modern approach to energy distribution, offering greater control and observability compared to traditional energy systems and promoting efficient energy use. By integrating internet technology, the smart grid enhances access to distributed and scalable energy resources on a vast scale [162]. The primary objective of these

innovations is to ensure consistent energy availability. However, the conventional smart grid, being centralized, is susceptible to single points of failure and potential data integrity issues at the central node. As a result, there's a growing shift towards decentralization in smart grid technology [163]. Looking ahead, the focus is on distributed energy sharing, which effectively manages energy generation and use among smart grid components [164]. But transitioning from a centralized to a decentralized smart grid introduces challenges, including transparency issues, high transaction costs, and delays in settlements, affecting throughput, scalability, and data security.

Blockchain technology addresses many of these challenges by enhancing the security of energy transactions through its immutable record-keeping, which ensures that transactions cannot be altered or deleted. It also secures communication between smart meters and the grid, allowing for transparent energy market operations and enabling peer-to-peer energy trading, thus reducing reliance on centralized energy providers and allowing consumers to participate directly in the energy market. However, blockchain-based smart grids face challenges related to transaction processing speed, particularly when dealing with high volumes of data and transactions, due to the latency introduced by existing blockchain consensus algorithms. These algorithms are resource-intensive, consuming significant energy during computation, which can be counterproductive in the context of smart grids focused on energy efficiency. To address these issues, advanced and secure ICTs like blockchain are pivotal in developing a sustainable distributed energy-sharing system. This has led us to introduce SmartGrid-NG, ETradeChain, and zk-SGB blockchain-driven smart grid systems designed for secure energy trading in next-generation smart grids. In this section, we have covered SmartGrid-NG and ETradeChain. The zk-SGB is covered in section 5.4.

5.3.1 Motivation and Contribution

Energy trading within smart grids is increasingly driven by the adoption of blockchain technology, which is motivated by its potential to enhance transparency, security, efficiency, and decentralization. With multiple parties involved in a decentralized energy market, ensuring every transaction is transparent is crucial. Blockchain's immutable ledger provides a clear record of all transactions, ensuring that every participant can verify and trust the recorded transactions without relying on intermediaries. As energy trading platforms become more digital, they also become targets for cyber-attacks. Blockchain's inherent cryptographic security measures and decentralized architecture make it resistant to a wide range of cyber threats. This shift towards a decentralized model promises to overcome the inefficiencies and vulnerabilities inherent in traditional centralized energy systems, paving the way for a more equitable and efficient distribution of energy resources. With this motivation, we have proposed two schemes for energy trading in the smart grid using blockchain technologies, which are SmartGrid-NG and ETradeChain. Specifically, our main contributions are as follows:

1. **Enhanced Security and Efficiency:** We introduce SmartGrid-NG, a system that seamlessly integrates smart grid technology with blockchain, aiming to strengthen the security and reliability of user transactions within the smart grid ecosystem. This integration is designed to eliminate vulnerabilities like single points of failure inherent in centralized systems. We have designed a consensus algorithm called GridChain to achieve efficiency and trustworthiness. This method employs a reputation-based leader selection scheme, ensuring that block generation and verification are both swift and entrusted to reliable nodes.

2. **Advanced Identity Authentication for Robust Security:** We have designed an efficient identity authentication scheme for the smart grid environment. This scheme prioritizes non-interactive key sharing between smart meters and consensus nodes, ensuring swift and secure communication. We have integrated the secure authentication MAC to strengthen data integrity further. For enhanced data confidentiality and a robust digital signature mechanism, we have incorporated elliptic curve cryptography (ECC) and ECDSA, respectively.

3. **New Cryptocurrency: Pcoin** In the proposed ETradeChain, we have introduced a novel energy cryptocurrency called Pcoin, which stands for Power Coin. Unlike existing cryptocurrencies such as Bitcoin, Ether, etc, Pcoins are associated with energy generated by any of the prosumers. Therefore, we have used Pcoins instead of an existing cryptocurrency. Further, the creation of Pcoins is completely dependent on the energy generated by the prosumers, unlike Bitcoin, Ether, and other cryptocurrencies, which are created based on high computing mining and reward. To create, transfer, and consume the Pcoins, there are three types of transactions, which are PcoinCreate, PcoinTransfer, and PcoinConsume.

4. **Fair Energy Trading and Prevention from Overpricing** Along with the new energy cryptocurrency in EtradeChain, we have addressed the issue of unfair mining and overpricing of energy using the modified double auction and Vickrey auction method. This combination helps the auction to increase its social welfare and also avoid overpaying. An optimal approach is proposed to allocate the consumer's bid to the prosumers. Before bid allocation, a novel sorting factor is added on the consumer end to avoid biased behaviour. During Bid allocation, thresholding is applied to sell the Pcoins so that starvation is avoided on the prosumer's end. The pricing calculation depends on Vickrey's auction and the mean bid prices. Also, the selection of temporary leaders is based on the prosumers who have won the auction.

5. **Comprehensive Transaction Security:** SmartGrid-NG and ETradeChain are both designed to be a beacon of transactional security, ensuring integrity, confidentiality, and authentication for every transaction. Beyond these foundational security measures, our proposed GridChain stands out by addressing and resolving prevalent blockchain challenges. These include ensuring fairness in transaction processing, safeguarding against double-spending attacks, and preventing forking in the chain, thereby maintaining the blockchain's integrity and trustworthiness.

6. **Performance and Security Analysis** We have measured the system's performance of SmartGrid-NG and ETradeChain based on parameters like consensus latency, throughput, computation overhead, energy consumption, and storage and communication overhead to show the effectiveness of this P2P energy trading system compared with state-of-the-art schemes. We compared the security analysis based on parameters like fork prevention, prevention of double-spending, and non-repudiation with state-of-the-art schemes.

5.3.2 Basic Terminologies in Smart Grid

There are some basic terminologies in smart grid that are important to understand when working with the proposed schemes. The basic terminologies are as follows.

- **Smart Meter:** The smart meter collects data on power consumption and real-time monitoring information of the consumer. They are installed at the consumer location. Every smart meter has a unique

ID, which is generated during the production of smart meters. When a smart meter is installed, this unique ID is registered in the blockchain via the consensus node. Any data recorded in the smart meter is considered a transaction, and these transactions are sent to the connected consensus node. These smart meters generate multidimensional data and send them to the connected consensus node. The connected consensus node may change if some fault occurs at the consensus node. If the connected consensus node goes down, then the smart meters connect to the nearest next consensus node. The smart meter is also connected to the smartphone to perform smart feedback to the grid operator. The Smart meters are also connected to the grid operator.

- **Consumer and Prosumer:** A consumer is typically defined as an individual or entity that purchases and consumes electricity. This includes residential households, commercial businesses, and industrial facilities. With the integration of blockchain technology and the evolution of smart grids, the role of consumers has expanded beyond traditional consumption. In a smart grid that utilizes blockchain, consumers often have the capability to generate their own electricity using renewable energy sources like solar panels or wind turbines. These consumers are then referred to as prosumers (producer-consumers). As prosumers, they not only consume the energy they need but can also sell any surplus energy back to the grid or directly to other consumers within the network. This is facilitated by blockchain technology, which provides a secure, transparent, and efficient platform for recording transactions and managing the flow of energy between different grid participants.

- **Grid Operator** The grid operator is the entity that is responsible for power distribution and sales of electricity in the connected region. The grid operator is connected to the region using the smart meter. Ensuring real-time monitoring and making a balance between demand and production are the main and important functions of the grid operator. The grid operator collects the consumer power consumption in real-time and maintains the demand for the region. This practice reduces the sudden pressure on the grid during peak hours, such as during the night hours, and enhances the reliability and robustness of the system. This practice also made the system fault-tolerant.

5.3.3 SmartGrid-NG: Blockchain Protocol for Secure Transaction Processing in Next Generation Smart Grid

SmartGrid-NG addresses the challenges highlighted above by leveraging blockchain's capability to store user information and transaction data securely. Traditional consensus algorithms such as PoS [6], PBFT [12], and PoW [1], fall short in the smart grid context. They are vulnerable to double-spending attacks, fork creation, and transaction integrity breaches and suffer from high consensus delays and low throughput, primarily due to their computational demands. Many existing methods also grapple with prolonged transaction processing times and significant computational, storage, and communication overheads. We introduced the GridChain consensus algorithm for SmartGrid-NG to overcome these limitations. The resultant SmartGrid-NG system represents a seamless fusion of smart grid technology and blockchain underpinned by the reputation-based GridChain consensus algorithm. We first explain the system architecture of SmartGrid-NG, then the methodology followed by the proposed novel consensus algorithm GridChain.

Smart Grid System Architecture

This section describes the general architecture of the SmartGrid-NG as shown in Figure 5.17. It has three components, i.e., Energy service provider (grid operator), permissioned blockchain network, and Consumer

Table 5.12: Mathematical symbol table for SmartGrid-NG and ETradeChain

Symbol	Meaning	Symbol	Meaning
TP	Transaction Pool	SD_i	i^{th} smart meter data
M_{ID}	Smart meter unique identity	T_i	Current Timestamp
Tr_i	i^{th} Transaction	δT	Threshold Time
SK_{pub}	Public Key of smart meter	SK_{pri}	Private key of smart meter
M_{pub}	Public Key of consensus node	M_{pri}	Private key of consensus node

(smart meter). All the components are connected, and relevant information is exchanged. The smart meters are installed at the place of the consumer and monitor the power consumption. The Grid operator is the service provider that distributes the power to the Consumer. The blockchain network is the permissioned blockchain network, where the miners are trusted and use the consensus algorithm for mining the blocks. In the proposed scheme SmartGrid-NG, we have proposed a consensus *Algorithm* 16 called GridChain. The blockchain network securely stores the user information and the transaction of the smart meter in the form of a blockchain.

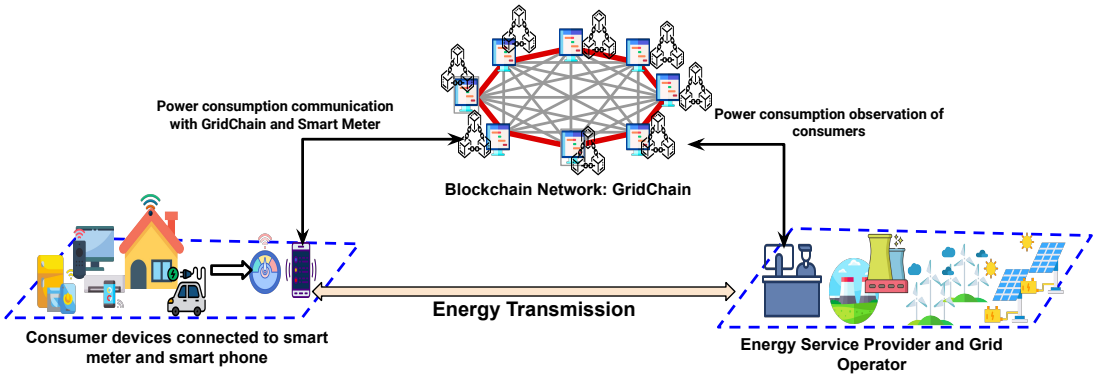


Figure 5.17: SmartGrid-NG system architecture

Methodology of SmartGrid-NG

This section presents the proposed consensus *Algorithm* GridChain and the framework SmartGrid-NG. As per the complete system execution, the process is divided into four phases: system initialization, secure transaction reporting, transaction extraction, and transaction verification and consensus. In the system initialization phase, all the entities of SmartGrid-NG generate secret parameters for secure communication of transactions. Further, all the consensus nodes are initialized with initial reputation values since the proposed Gridchain is a reputation value-based leader selection scheme to reach consensus for block generation and verification. The next two phases (transaction reporting and transaction extraction and verification) are the secure and authentic communication of transactions between the smart meter and the consensus nodes. In the fourth phase, the consensus is responsible for selecting the leader node and creating a blockchain ledger of authentic transactions. The symbol Table 5.12 defined various symbols that are used in the proposed schemes.

Phase 1: System Initialization

In this phase, the smart meter or energy consumer, grid operator, and consensus nodes generate the public and private Keypair-based elliptic curve cryptography method. Also, we have used the secure hash and authentication function to authenticate the smart meter and the consensus nodes.

1. Let $q > 3$ be a prime number. An equation of the form defines an elliptic curve E over F_q (finite field)

$$y^2 = x^3 + ax + b \tag{5.11}$$

where $a, b \in F_q$ and $4a^3 + 27b^2 \neq 0$ (mod q). The set $E(F_q)$ consists of all points $(x, y), x \in F_q, y \in F_q$, which satisfy the defining equation (5.11). In ECC keypair generation, each entity A does the following:

   (a) Select a random or pseudo random integer d in the interval $[1, n-1]$ (here n is an finite integer number).

   (b) Compute $Q = d * G$. (here, G is the generator polynomial point on the ECC curve defined by equation (5.11)).

   (c) A's public key is Q; A's private key is d.

2. Execute the following computation to assign the secret information of all the smart meters $SM = \{SM_1, SM_2, ...SM_N\}$ and consensus node $CN = \{CN_1, CN_2, ...CN_M\}$.

   (a) Randomly select $Pr_{SM_i} \in F_q$ and calculate $Pb_{SM_i} = Pr_{SM_i} * G$. Pr_{SM_i} and Pb_{SM_i} are the private and public keys of SM_i with identity ID_{SM_i}, respectively.

   (b) Randomly select $Pr_{CN_j} \in F_q$ and calculate $Pb_{CN_j} = Pr_{CN_j} * G$. Pr_{CN_j} and Pb_{CN_j} are the private and public keys of CN_j with identity ID_{CN_j}, respectively.

   (c) Select one-way secure hash function $H : 0, 1^*$, which is SHA256.

   (d) Initialize the secure authentication function, i.e., message authentication code (MAC) as $MAC_{sk} = HMAC(sk, Tr)$, where sk is the shared secret key, and Tr is the transaction data.

   (e) Publish the public system parameters $< E, G, q, H() >$, $< ID_{SM_i}, Pb_{SM_i} >$, $< ID_{CN_j}, Pb_{CN_j} >$, where $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, M$ amongst the consensus nodes.

3. Based on the performance and status of each consensus node CN_j is initialized with a reputation value $Rep_{CN_j} \in [01, 1]$, $\forall j, j = 1, 2, ..., M$, and the reputation threshold is

$$Rep = \frac{\sum_{j=1}^{M} Rep_{CN_j}}{M} \tag{5.12}$$

Phase 2: Transaction Reporting

In this phase, the SM_i generates the transactions and reports them to the consensus node CN_j, $\forall j, j = 1, 2, ..M$. The SM_i performs the following computation.

1. Transactions generated at time $t_\tau < tr_1, tr_2, ...tr_k >, k = 1, 2, ..P$.

2. Encode the transactions $Tr_k = H(tr_k||ID_{SM_i}||ID_{CN_j}||t_\tau)$

3. Calculate the shared session key with CN_j
   $Sk_i^{CN_j} = Pr_{SM_i} * Pb_{CN_j}$.

4. To encrypt and send the transactions to CN_j,

   (a) SM_i select a random positive integer k.

   (b) SM_i then generate the ciphertext CT_{Tr_k}
   $CT_{Tr_k} = \{k * G, Tr_k + k * Sk_i^{CN_j}\}$. SM_i used the shared secret key for the encryption.

5. Calculate $MAC(Sk_i^{CN_j}) = HMAC(Sk_i^{CN_j}, < Tr_k, t_\tau >)$.

6. send $< CT_{Tr_k}, MAC(Sk_i^{CN_j}), t_\tau >$ to CN_j.

Phase 3: Transaction Extraction and Verification

In this phase, the CN_j first generates the shared session key with SM_i, then extracts the transaction from received ciphertext CT_{Tr_k}. It also verifies the authenticity of received information by comparing the received MAC and calculated MAC.

1. Calculate the shared session key with SM_i
   $Sk_{CN_j}^i = Pr_{CN_j} * Pb_{SM_i}$.

2. To decrypt and receive information from SM_i.

   (a) SM_i select a random positive integer k.

   (b) SM_i then generate the ciphertext CT_{Tr_k}

3. Calculate $MAC(Sk_{CN_j}^i) = HMAC(Sk_{CN_j}^i, < Tr_k, t_\tau >)$. if it equals to $MAC(Sk_i^{CN_j})$, then the transaction is accepted otherwise rejected. It is possible because both the shared secret keys are equal, i.e., $Sk_i^{CN_j}$ is equal to $Sk_{CN_j}^i$.

4. calculate $\sigma_{CN_j}^{(1)} = H(Tr_k)^{Pr_{CN_j}}$, where $\sigma_{CN_j}^{(1)}$ is the signature of Tr_k.

5. send $< Tr_k, \sigma_{CN_j}^{(1)} >$ to CN_x. CN_x is the leader node of the next round of consensus process. In the next section, we have defined how the leader node is selected.

GridChain: Reputation based Consensus

This phase is responsible for the blockchain-specific task, that is, block generation and verification, and adding this block into the blockchain ledger. To accomplish this task, we proposed a GridChain blockchain protocol, which is a leader-based consensus algorithm, and the leader node is selected based on the corresponding reputation value. Let, we have a set of consensus nodes $CN_j = \{CN_1, CN_2, ..., CN_M\}$, where $j = 1, 2, ..., M$. Among these nodes, the CN_x, $x \in j$ is the leader node, which receives the information from

the smart meter SM_i. The whole consensus process is divided into three phases, i.e., 1. leader node selection, 2. block generation and verification, and 3. node reputation updation.

1. Leader Node Selection In each round of the consensus process, a leader node is selected. To make the leader selection process more secure and fair, first, the consensus nodes (CN) whose reputation value is above or equal to the reputation threshold are selected to form the pre-selected group. Second, to make the process fair or prevent the leader node from being known in advance by the attacker, the nodes that satisfy the random number threshold are selected from the pre-selected group. Finally, from these pre-selected consensus nodes, a node with a high reputation value is selected as the leader node. Thus, the leader selection process is a three-step process, i.e., create a pre-selection group, random reputation threshold, and find the leader node.

1. **Create Pre-selection Group**: There are two possible cases.
   (a) **Case 1: If it is the first round of consensus** In this case, the consensus nodes CN_j with $Rep_{CN_j} \geq Rep$ are selected to form the pre-selection group. Here, the Rep_{CN_j} is the node's initial reputation value, and Rep is the reputation threshold value.
   (b) **Case 2: Otherwise** In this case the consensus nodes CN_j with $Rep^*_{CN_j} \geq Rep$ are selected to form the pre-selection group. Here, $Rep^*_{CN_j}$ is the updated reputation value of the node CN_j for $j = 1, 2, ..., M$.
   (c) Let, there are $S \in M$ number of consensus nodes in the pre-selection group, i.e., $CN_s = \{CN_1, CN_2, ..., CN_s\}$, $s = 1, 2, ..., S$.
2. **Random Number Threshold** Consensus nodes CN_s in the $S \in M$ pre-selection group perform the following computation.
   (a) Select the random number X_s.
   (b) Sign the X_s using ECDSA as $\sigma_{CN_s} = H(X_s)^{Pr_{CN_s}}$, where, Pr_{CN_s} is the private key of CN_s.
   (c) Broadcast this into the other nodes in the pre-selection group.
3. The current leader node CN_x performs the following steps to generate the random number threshold.
   (a) Select the random number X_{CN_x} as the threshold ω.
   (b) Sign the X_{CN_x} using ECDSA as $\sigma_{CN_x} = H(X_{CN_x})^{Pr_{CN_x}}$, where Pr_{CN_x} is the private key of the leader node CN_x.
   (c) Send $< \omega, \sigma_{CN_x} >$ to each node CN_s.
   (d) Consensus nodes CN_s with $X_s \geq \omega$ form the group $Q \in S$.
4. Among the consensus nodes group Q, the node with the highest reputation value $Rep^*_{CN_j}$ is selected as the leader node CN_x.
5. The information of CN_x is broadcasted to each consensus node group CN_j, $j = 1, 2, ..., M$.

2. Block Generation and Verification The block verification scheme is also simple yet very effective. The newly elected leader node sends the newly created block along with the nonce value. Every verification node again creates the block with the received nonce value and checks the received block's validity. Therefore, the verification process also does not take much time to block verification. These are the two novel

things that are introduced in the GridChain consensus algorithm to create, verify, and add the block to the distributed ledger. We have considered that every miner node has a transaction pool, where the smart meters of the consumers put all the transactions. The miner node sends the transaction to other miner nodes after the transactions are verified. The consensus *Algorithm* 16 works as follows:

Algorithm 16: Block Generation and Verification

Input: $Tr_i : Tr_1, Tr_2, \ldots, Tr_n$
Output: *Block*
$TP.append(Tr_i)$;
Function `blockGenerate(`Tr_{New}`):`
  if *node == leader* **then**
    $Mrt = createHashTree(TP.getAllTransactions)$;
    $PBH = H(PB)$;
    $BH = H(T, PBH, Mrt)$;
    $block = createBlock(BH, PBH, T, n, Mrt)$;
    $broadcast(sign(block, nonce))$;
Function `blockVerification(`*block, nonce*`):`
  if *node != leader* **then**
    $getBlock(block, nonce)$;
    $Mrt = createHashTree(TP.getAllTransactions)$;
    $block* = createBlock(Mrt, T, PBH, BH, n)$;
    if *block == block** **then**
      setBlockStatus(valid);
    else
      setBlockStatus(invalid);
      node.status = bad;
  if $(count(block_status == valid) \geq N/2+1)$ **then**
    blockchain.append(block);

1. The smart meter (SM_i) is considered an ordinary node that is responsible for generating the transactions. All the generated transactions are put into the transaction pool TP. The selected leader node collects the transactions from the TP for verification. After verification, it transfers those transactions to other miner nodes. Then, the leader node calls the function *createHashTree* to calculate the Mrt of these transactions. The hash PBH of the previous block is calculated with the function $H()$, and the current timestamps T and *nonce* are set to the appropriate value.

2. Using the values of $Mrt, PBH, T, nonce$, the leader node creates the block using $createBlock()$ function. The leader node sends the recently mined block to all the nodes for verification. The digital signature is used to prevent the block from modification and impersonation attacks.

3. When the rest of the nodes (other than the leader node) get the copy of the block, they first verify the signature of the block to check whether it was sent from the legitimate leader node or not. After validating, they also verified the correctness of the block. To validate the block, the non-leader nodes recreate the block from the same set of transactions, and for that, it calculates the $Mrt, PBH, T, nonce$.

The block is created using the same set of transactions and other required data, and then non-leader nodes match this with the received block. If a match is found, the node updates *block_status* to *Valid*; otherwise, it updates to *Invalid*. In addition to *block_status*, the node will mark the leader node's status (*node.status*) as bad for future leader selection.

4. Assume the leader node gets at least $\frac{N}{2}+1$ responses as *Valid*. In that case, it appends the block in the distributed ledger, i.e., a block is considered valid when half of the nodes verify the block. At least 51% nodes must verify the block to prevent any attack.

3. Node Reputation Updation Reputation is the continuous evaluation of the long-term behaviour of consensus nodes. Based on the subjective logic theory in [158], we derive our reputation model. Lets, assume the consensus node CN_i interact with CN_j of each round of consensus, where $i = 1,2,...,M$, $j = 1,2,...,M$, and $i \neq j$. Based on the interaction between both the nodes, CN_i calculates reputation $Rep^k_{i,j}$ of CN_j in the k^{th} round of the consensus process through the subjective logical reputation model. In the last step, the current leader node CN_x obtain the final reputation value $Rep^*_{CN_j,CN_i}$ of CN_j. To accomplish this, it integrate the $Rep^k_{i,j}$ and $Rep^{*(k-1)}_{CN_j}$, where $Rep^{*(k-1)}_{CN_j}$ is the reputation value of CN_j in the $(k-1)^{th}$ round.

The reputation value of $Rep^*_{CN_j,CN_i}$ is computed between the nodes CN_i and CN_j based on their interaction during the consensus process. In the consensus process, the block verifier node also responds to the node status as good or bad. This can be divided into two steps, i.e., opinion calculation and integration with past reputation values.

1. **Opinion Calculation:** Based on the interaction behavior between CN_i and CN_j, we calculate trust ($T_{i,j}$) and uncertainty ($U_{i,j}$), respectively, where $i = 1,2,...,M$, $j = 1,2,...,M$, and $i \neq j$. The opinion calculation of CN_i to CN_j is calculated based on the $T_{i,j}$ and $U_{i,j}$. The computation is as follows:

   (a) **Interaction Behavior:** As shown in the consensus *Algorithm* 16, the verifier node also responds to the status of the leader node. Therefore, interaction behaviour between CN_i and CN_j is of two types, i.e., positive interaction behaviour depicted as $\mathbb{P}$ and negative interaction behaviour depicted as $\mathbb{N}$. This can be quantified as depicted as $\mathbb{P}+\mathbb{N}=1$, and $\mathbb{P}>\mathbb{N}$.

      i. **Positive Interaction Behavior:** If the consensus node CN_i sends some information to CN_j, then this interaction is considered a positive interaction if 1) the signature of the received information is successfully verified and 2) it is a faster response time. Let's say the response timeline is T_τ, and the active timeline is T_0, and if $T_\tau < T_0$, then it is considered a faster response time. Assume δ and ξ are the fast and slow response times, respectively, then $\delta+\xi=1$, and $\delta>\xi$.

      ii. **Negative Interaction Behavior:** The negative interaction between any two consensus nodes CN_i and CN_j can be quantified based on the two cases: 1) wrong signature and 2) sending delayed signature. The quantification can be expressed as $\lambda+\eta=1$,$\lambda>\eta$, where λ is depicting the wrong signature and η is depicting as sending a delayed signature.

   (b) **Effectiveness of Recent Interaction:** It is important to note that the recent interaction between CN_i and CN_j has a greater effect on the reputation evaluation than the past interaction. The quantification of this interaction can be expressed as $\rho+\phi=1$, $\rho>\phi$, where ρ is the recent interaction and ϕ is the past interaction.

Based on the above evaluation parameters, the trust and uncertainty can be computed as follows:

$$T_{i,j} = \frac{S_{i,j}.\alpha_{i,j}}{\alpha_{i,j} + \beta_{i,j}} \tag{5.13}$$

$$\alpha_{i,j} = \mathbb{P}[\rho(\delta\alpha_{1,1} + \xi\alpha_{1,2}) + \phi(\delta\alpha_{2,1} + \xi\alpha_{2,2})] \tag{5.14}$$

$$\beta_{i,j} = \mathbb{N}[\rho(\lambda\beta_{1,1} + \eta\beta_{1,2}) + \phi(\lambda\beta_{2,1} + \eta\beta_{2,2})] \tag{5.15}$$

$$U_{i,j} = 1 - S_{i,j} \tag{5.16}$$

Here, $S_{i,j}$ is the probability of successful interaction between CN_i and CN_j, where $i \neq j$. $\alpha_{1,1}$ and $\alpha_{1,2}$ are the recent interactions with fast/slow response time in positive interaction. $\alpha_{2,1}$ and $\alpha_{2,2}$ are the past interactions with fast/slow response time in positive interaction. $\beta_{1,1}$ and $\beta_{1,2}$ are the recent/past interactions with the wrong signature in negative interaction. $\beta_{2,1}$ and $\beta_{2,2}$ are the recent/past interactions with delayed signatures in negative interaction. Based on the above computation, the opinion calculation can be expressed as

$$Rep^{*(k)}_{CN_i,CN_j} = T_{i,j} + Rep^{*(k-1)}_{CN_j}.U_{i,j} \tag{5.17}$$

Computing Reputation Value: When the above computation is completed, the current leader node CN_x computes the final reputation value of CN_j as follows.

1. The consensus node CN_i send the reputation value $Rep^{*(k)}_{CN_i,CN_j}$ which is the local reputation value of CN_i to CN_j based on the opinion behavior in the k^{th} round of the consensus process.
2. The current leader node prepares the 2X2 reputation matrix as follows:

$$\begin{bmatrix} Rep^{*(k)}_{CN_1,CN_j} & , & Rep^{*(k-1)}_{CN_1} \\ Rep^{*(k)}_{CN_2,CN_j} & , & Rep^{*(k-1)}_{CN_2} \\ & \vdots & \\ Rep^{*(k)}_{CN_M,CN_j} & , & Rep^{*(k-1)}_{CN_M} \end{bmatrix}$$

Here, $Rep^{*(k)}_{CN_1,CN_j}$ depicts the reputation value of CN_j on CN_i in the k_{th} round of consensus, and $Rep^{*(k-1)}_{CN_j}$ is the reputation value of CN_j in the $(k-1)^{th}$ consensus round.

The Current leader node CN_x updates the final reputation value of CN_j in the k_{th} round of consensus based on the above computation as follows.

$$Rep^{(final,k)}_{CN_j} = \frac{Rep^{*(k)}_{CN_1,CN_j} Rep^{*(k-1)}_{CN_1} + ... + Rep^{*(k)}_{CN_M,CN_j} Rep^{*(k-1)}_{CN_M}}{\sum_{j=1}^{M} Rep^{*(k-1)}_{CN_j}} \tag{5.18}$$

Reward Mechanism in SmartGrid-NG

leader node and the verifier consensus nodes get the reward for mining the block in GridChain. The remuneration or reward is provided based on the blocks mined by the leader node, i.e., each block mining

contains a fee, which is referred to as a transaction fee. Instead of using Bitcoin as a reward, GridChain uses a transaction fee as a recompense, which means that every transaction has a transaction cost. If every block contains N transactions, then the general equation for total transaction fee (TTF) equals to

$$TTF = \sum_{i=1}^{N} [TF_i] \tag{5.19}$$

where TF is a transaction fee, and N is the number of transactions in the j^{th} block. This transaction fee is divided into two parts, where 60% is sent to the leader node (CN_x) and the 40% is equally distributed to the honest transaction verifier consensus nodes (CN_j^{honest}, $j = 1, 2, ..., S$). Here, S is the number of consensus nodes in the pre-selection group. This can be expressed as

$$< CN_j^{honest}, TTF_{CN_j} > \leftarrow \frac{40\%(TTF)}{S} \tag{5.20}$$

$$< CN_x^{honest}, TTF_{CN_x} > \leftarrow 60\%(TTF) \tag{5.21}$$

For example, let the leader node mine block where the block contains 516 transactions, and the transaction fee is 0.1$ per transaction, then the TTF would be as $TTF = \sum_{i=1}^{516}[0.1]$, $TTF = 51.6$\$. This is how the remuneration of the miner node is done, and it gets benefits for the transaction processing and mining of the block. The equal distribution of 40% of block mining reward in the transaction verifier consensus nodes is very important To encourage and get equal and honest participation from them. Additionally, it is also important to avoid malicious block generation possibilities from the leader nodes. Therefore, due to these challenges, we have divided the block mining reward and equally distributed it amongst the honest verifier consensus nodes.

5.3.4 ETradeChain: Blockchain-Based Energy Trading in Local Energy Market (LEM) using Modified Double Auction Protocol

The proposed ETradeChain is a next-gen smart grid technology that resolves issues in current methods. It's a concurrent blockchain-based energy trading system ideal for decentralized LEM. This concurrent approach offers better efficiency and higher throughput than traditional blockchain for smart grids. In this permissionless network, energy is traded for smart homes using "Pcoins" (Power Coins), which represent energy generation. Prosumers can exchange generated energy for Pcoins, and others can buy these Pcoins to get energy. An auction mechanism is essential for fair energy trading and allocation. An auction mechanism is a great way to manage and allocate energy. Auctions have been widely researched for trading in areas like cloud virtual machines [134] [132], edge computing resources [134], [165], Smart Grids [166] [133] [167] [168]. Most auction systems rely on a central authority to oversee the process, and some are computationally intensive, making them unsuitable for devices like smart meters. Many are also unfair and suboptimal. In contrast, ETradeChain offers a lightweight, fair, and decentralized auction mechanism. This auction also selects temporary leaders for consensus in block creation and verification.

Goal of ETradeChain

ETradeChain is a decentralized P2P network with N mining nodes. Each node uses a key pair to sign and verify blocks and transactions, eliminating the need for a central PKI. Transactions are cryptographically hashed using function $H(x)$ before block inclusion. At the time t, some nodes, $M(t) \in N$, are Byzantine

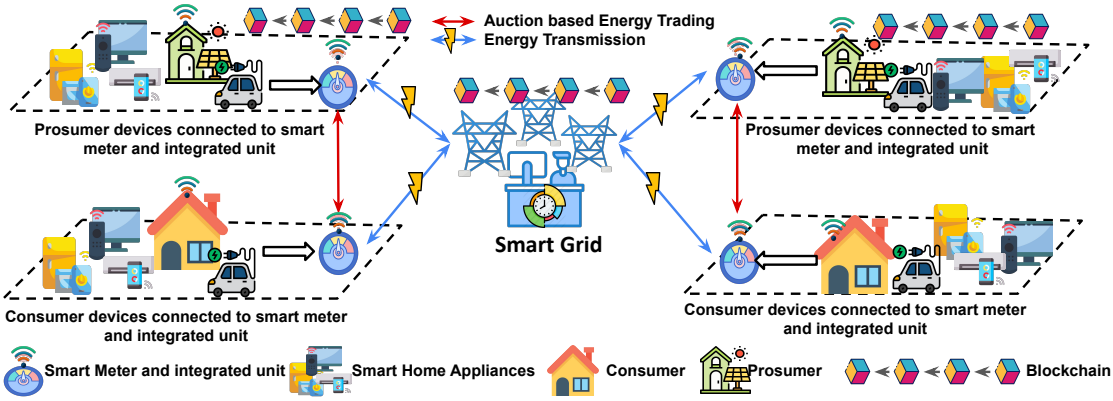


Figure 5.18: ETradeChain system model

and controlled by an adversary, while others are honest. Given a node's mining power as $Mining(K)$, the combined power of Byzantine nodes must be under 51% of the total N nodes' power. This is because ETradeChain included all such vulnerable nodes, which do selfish mining, under the control of the adversary.

$$\sum_{k \in M(t)} Mining(k) < 51\% \sum_{i \in N} Mining(i) : \forall K, i \tag{5.22}$$

System Model of ETradeChain

ETradeChain's architecture comprises three components: Smart Grid (Auctioneer), prosumer (Seller + Buyer), and consumer (Buyer), depicted in Figure 5.18. A LEM Testbed is set up using Raspberry Pi boards, which act as smart meters with integrated energy trading units. These meters, placed at consumer locations, track power consumption. The Smart Grid distributes power to consumers. Prosumers create a permissionless blockchain network, which securely logs user details and smart meter transactions in a ledger.

There's a consistent market-clearing price for each time slot $t \in T$. Customers with full bid orders pay this price for the specified power amount. Blockchain facilitates both market operations and payments. Smart meters automatically measure and predict each customer's demand d_{it} and generation g_{it}. Based on this, any excess demand or supply is computed and credited to the customer's blockchain account, ensuring data privacy while aligning with consumption patterns. Accounts hold Pcoin balances and energy data. By combining a customer's energy surplus and credit balance, orders are tailored to their needs. This virtual trading doesn't affect actual electricity transfers. We use a supply and demand matching online mechanism and price signals to indicate the local scarcity of energy. As a result, the large-scale power grid can be accessed. A forecast of consumption and production at the customer level is used to match the LEM's demand and supply restrictions. We only include electricity trading and ignore bundled orders (like electricity and heat) to keep things simple. The LEM's result is calculated as the total cost of electricity for all customers. In other words, this is the total amount of electricity x_{ijt} sold from one customer i to another customer j at a price c_{ijt} at time t. The minimum cost of the energy sold from prosumer i to prosumer/consumer j is represented in Equation (5.23). Equation (5.24) ensures that every prosumer satisfies their demand d_{it} and sells the surplus generation g_{it}. Equation (5.25) and (5.26) ensure that the

smart grid balances supply and demand within the LEM. The hard constraints for this are the equations from (5.23) to (5.26). The decision variable x_{ijt} has to be positive because only a positive amount of electricity can be traded.

$$min \sum_{i \in I, j \in J} c_{ijt} * x_{ijt} \tag{5.23}$$

$$g_{it} - d_{it} - \sum_{j \in I} (x_{ijt} - x_{jit}) = 0, \forall i \in I \tag{5.24}$$

$$\sum (g_{it} - d_{it} \leq \sum_{i \in I} X_{igt}) \tag{5.25}$$

$$-(\sum_{i \in I} (g_{it} - d_{it})) \leq \sum_{i \in I} x_{git} \tag{5.26}$$

Problem Definition of Energy Trading in ETradeChain

Let us define and formulate our problem. We consider that we have N prosumers and M consumers. Let P define the set of prosumers, and C define the set of consumers such that $P = \{1,2,3...,N\}$ and $C = \{1,2,3...,M\}$. Any prosumer, while bidding, will have two factors ($bidP_j$, s_j). $bidP_j$ denotes the selling price of one Pcoin as a bid raised by j^{th} prosumer and s_j denotes the total number of Pcoins j^{th} prosumer wants to sell. Similarly, consumers also have two factors ($bidC_i$, d_i). $bidC_i$ denotes the bidding price of one Pcoin raised by i^{th} consumer and d_i denotes the total Pcoins demand of i^{th} consumer. The utility of i^{th} consumer will be $u(C_i) = d_i \cdot bidC_i - d_{si} \cdot avg(C_i)$ where the $avg(C_i)$ denotes the average amount i^{th} consumer has to pay for one Pcoin to the seller and d_{si} denotes the total Pcoins received by i^{th} consumer. Similarly, the utility of j^{th} prosumer is equal to $u(P_j) = s_{sj} \cdot act(P_j) - s_j \cdot bidP_j$ where $act(P_j)$ denotes the actual amount received by the prosumer for the Pcoins and s_{sj} denotes the total Pcoins received by the j^{th} prosumer. Therefore, the total Utility of the auction can be defined as follows

$$U = \sum_{j=1}^{N} u(P_j) + \sum_{i=1}^{M} u(C_i) \tag{5.27}$$

We are considering the auction process to be balanced, i.e., the total amount received by the prosumer should be equal to the total amount given by the seller at the end of the auction process. Therefore $\sum_{j=1}^{N} s_{sj} \cdot act(P_j) = \sum_{i=1}^{M} d_{si} \cdot avg(C_i)$

$$U = \sum_{j=1}^{N} s_j \cdot bidP_j - \sum_{i=1}^{M} d_i \cdot bidC_i \tag{5.28}$$

Therefore, we can define our problem as follows.

Maximise $U = \sum_{j=1}^{N} s_j \cdot bidP_j - \sum_{i=1}^{M} d_i \cdot bidC_i$

Subject to:

$$0 \leq s_{sj} \leq s_j, \forall j \in N \tag{5.29}$$

$$0 \leq d_{si} \leq d_i, \forall i \in M \tag{5.30}$$

$$\sum_i d_{si} = \sum_j s_{sj}, \forall i \in M, \forall j \in N \tag{5.31}$$

$$d_i, s_j > 0, \forall i \in M, \forall j \in N \tag{5.32}$$

According to equation (5.29), the total number of Pcoins sold by the prosumer should be less than or equal to the total number of Pcoins available with the prosumer to sell. Equation (5.30) tells that the total number of Pcoins received by the consumer should be less than or equal to the Pcoins requested. Equation (5.31) defines the total Pcoins sold by the prosumer should be equal to the total Pcoins received by the consumer. Further, Equation (5.32) explains the total Pcoins bid by both prosumer and consumer should be greater than 0.

Methodology of ETradeChain

This section explains the protocol in detail. ETradeChain is a blockchain-based Smart Grid system for processing transactions. Here, we have considered three types of transactions: PcoinCreate, PcoinTransfer, and PcoinConsume. The detail of these transactions is given in the subsequent section. All the energy trading happens with the use of a modified double auction scheme. The winners of the auction are selected as temporary leaders. These leaders create the blockchain network for the consensus on block creation. In this network, one super leader is elected. The super leader selection is based on the stake deposited by the leader nodes. The Pcoins are used as a stake. The leader node that deposited the maximum stake will have the highest probability of being selected as the super leader node. The super leader node is responsible for block creation and transaction verification. However, transactions are verified by the other leader nodes.

A. Type of Transactions

The energy is traded between the nodes via a smart grid energy distributor in the LEM. Three types of transactions are generated for trading energy, which are described below.

1. **PcoinCreate**: Pcoins are created by prosumers based on the generated electricity. The excess electricity is given to the power grid, which generates the equivalent coins. With respect to this coin generation, the transaction is generated, which is called the PcoinCreate transaction.

2. **PcoinTransfer**: The consumer who wants to get electricity from the smart grid first needs to buy some Pcoins. This happens via the auction process. After the Bid is allocated and the pricing of Pcoins is calculated between the prosumer and consumer, the Pcoins and money are transferred between them. This type of transaction is called "PcoinTransfer."

3. **PcoinConsume**: The consumer, or prosumer, can get electricity from the smart grid in exchange for Pcoins, and this transaction is called a "PcoinConsume" transaction. The Pcoins that are consumed will never be used again to get electricity.

B. Auction Mechanism

Below is the process of a modified double auction mechanism where both consumers and prosumers can bid to buy Pcoins and ask to sell Pcoins. Further, we are going to use the Vickrey auction or second-price auction to allow for truthful action. Whenever any prosumer wants to participate in the trading process, they must first register at the smart grid (certificate authority) for generating key pairs. After getting the key pair, the public key acts as an identity for the node. The auction is done so that prosumers can sell their Pcoins and consumers can buy those Pcoins. There are three important steps in the complete auction process.

- *Bid*: It is the first step in the auction process. It has a dedicated time phase during which prosumer and consumer share their bid with the smart grid. After the completion of this phase, the smart grid broadcasts all the collected bids to the peers in the network.
- *Allocate*: After receiving the smart grid's bid details, all the blockchain nodes calculate the bid allocation and pricing scheme and share the final output with the peers. Further, all the nodes verify the schemes and reach on consensus using the voting mechanism.
- *Energy Exchange*: Once the above steps are completed, the consumers can get their energy in exchange for Pcoins from the smart grid by invoking the PcoinConsume transaction.

C. Bidding Process

The bidding process initiates the auction mechanism. This bidding process is executed in a dedicated time phase or a slot. During this phase, all the consumers and prosumers who want to take part in the auction will share their bids with the smart grid. The consumer also shares the demand for the Pcoins that it requires, and the prosumers will share the number of Pcoins that it wants to sell. Once this phase of the bidding process is over, the Smart grid will then broadcast the collected bids, consumer demand, and prosumer supply details with the peers in the network. The peers then use these details for the next steps.

D. Sorting Mechanism

In our case, the sorting mechanism is done off-chain, all the nodes conduct the auction mechanism, and the final allocation and pricing scheme is broadcasted to other nodes, unlike the central authority or auctioneer performing the Auction mechanism as per in [166] [167]. The allocation and pricing scheme, which has been taken by 50% of the nodes in the network, is chosen. Further, [167] tries to sort the buyers (consumers) in descending order on the basis of their bid price. But on the consumer end, there is a possibility that a single buyer bids the highest for a large amount of Pcoins, so to avoid that, we created a sorting factor for the consumer as follows

$$sf(C_i) = \frac{bidC_i}{\sqrt{d_i}} \tag{5.33}$$

From the above equation, it can be seen that the sorting factor doesn't only depend upon the bid price but is inversely proportional to the square root of the Pcoins demanded. This keeps a single buyer from buying all the Pcoins by paying a high price. After this, we sort this Sorting Factor of consumers in descending order. In the case of the prosumers, we are trying to sort them in ascending order of their bid prices.

E. Bid Allocation Mechanism

We have implemented an optimal approach to allocating the consumer's bid to the prosumers. To avoid starvation, we are creating a threshold value on the number of Pcoins a prosumer could sell to a particular consumer. If consumer demand is not satisfied by the first prosumer, his demand will be satisfied by the next prosumer in the sorted order, and so on. The first prosumer, after satisfying one consumer's demands, will satisfy the next consumer in reverse sorted order of the sorting factor. Below is the formulation of the threshold value.

$$Th_s = \frac{\sum_{j=1}^{N} s_j}{M \cdot N} \tag{5.34}$$

In the *Algorithm* 17, we are considering that the consumers and prosumers are already sorted as per the sorting mechanism discussed above, and C_1 and C_M are the consumers having the highest and lowest sorting factors, respectively, and also P_1 and P_N are the prosumers having the lowest and highest bid price, respectively. The algorithm for bid allocation works as follows:

1. After prosumers and consumers are sorted as per the sorting mechanism, a threshold is calculated using (5.34) and mapping is done between the consumers and the prosumers. The allocation matrix (AM) stores the number of Pcoins that need to be transferred from j^{th} prosumer to the i^{th} consumer.
2. The following steps are executed only if the bid amount of consumers is greater than the bid amount of prosumers. Step-1: If the d_i of the consumer is satisfied, it goes for the i+1 consumer. Step 2: If the j^{th} prosumer has no more Pcoins, then go for the j+1 prosumer
3. If the d_i is pending and greater than or equal to the threshold value and also if the j^{th} prosumer has greater than or equal to the th_s value of Pcoin then Increment the allocation matrix of i^{th} consumer and j^{th} prosumer by threshold and decrement the s_j and d_i by threshold.
4. If the d_i is pending and less than the threshold value and also if the j^{th} prosumer has greater than or equal to the requested value of Pcoin by i^{th} consumer, then Increment the allocation matrix of i^{th} consumer and j^{th} prosumer by d_i and decrement the s_j by d_i and make d_i as 0.
5. If the j^{th} prosumer has less Pcoins than the d_i or th_s then Increment the allocation matrix of i^{th} consumer and j^{th} prosumer by s_j and decrement the d_i by the s_j and make s_j as 0.

F. Pricing Mechanism

After the bid allocation is done, the next step is to create the pricing mechanism. We make use of the mean of consumer and prosumer bid prices and the Vickrey Auction to calculate the pricing mechanism. A Vickrey or second-price auction prevents the consumer from overpaying, improves the competition, and optimizes future bids.

Suppose the i^{th} consumer receives x Pcoin from the j^{th} prosumer. Below is the amount the i^{th} consumer needs to pay to the j^{th} consumer.

$$pay_{ij} = x \cdot (\frac{bidP_j + bidC_{i+1}}{2}) \tag{5.35}$$

For the last consumer, the bid price paid will be the amount they bid on only. As per the algorithm 18, a pay matrix (PM) is created, which contains the payment mapping of i^{th} consumer to the j^{th} prosumer. The

Algorithm 17: Bid Allocation Mechanism

Input: M Consumers, N Prosumers, d_i is the number of Pcoin i^{th} consumer (C_i) requires, s_j is the number of Pcoin j^{th} prosumer (P_j) wants to sell

Output: Allocation Matrix (AM)

Function `generateAM`$(M, N, bidC_i, bidP_j, s_j, d_i)$**:**

$Th_s = \frac{\sum_{j=1}^{N} s_j}{M \cdot N}$;
for $i \leftarrow 1$ **to** M **do**
  for $j \leftarrow 1$ **to** N **do**
    if $bidC_i \geq bidP_j$ **then**
      if $d_i == 0$ **then**
        break;
      else if $s_j == 0$ **then**
        continue;
      else if $d_i \geq Th_s$ ***and*** $s_j \geq Th_s$ **then**
        $AM_{ij} += Th_s$;
        $d_i -= Th_s$;
        $s_j -= Th_s$;
      else if $d_i < Th_s$ ***and*** $s_j \geq d_i$ **then**
        $AM_{ij} += d_i$;
        $s_j -= d_i$; $d_i = 0$;
      else
        $AM_{ij} += s_j$;
        $d_i -= s_j$;
        $s_j = 0$;

pay matrix is updated with the help of (5.35) except for the last consumer, which pays the price bided by it. From the algorithm, it can be seen that the total payment received by the j^{th} prosumer is equal to the sum of the j^{th} column in the pay matrix. Also, the total payment given by the i^{th} consumer is equal to the sum of the i^{th} row in the pay matrix.

G. Temporary Leader Selection From Auction

Miner nodes are temporary leaders chosen from among all prosumers. One of these miner nodes is chosen as the super leader, who creates the block, and the other mining node is chosen as the verifier. The winners of the auction system are used to determine the temporary leaders. It is based on the concept of the original double auction method. The goal of selecting the auction scheme leaders is to involve more prosumers and consumers in trade. The procedure for selecting a temporary leader through an auction is described below.

1. Sort the Consumers in decreasing order of their bid: $bidC_1 \geq bidC_2 \geq ... \geq bidC_m$.
2. Sort the Prosumers in increasing order of their bid: $bidP_1 \leq bidP_2 \leq ... \leq bidP_n$.
3. Let k be the largest "breakeven index" index such that $bidC_k \geq bidP_k$

Algorithm 18: Pricing Mechanism

Input: M Consumers, N Prosumers, AM Allocation Matrix, $bidC_i$ is the bid price of one Pcoin of i^{th} consumer (C_i), $bidP_j$ is the ask price of one Pcoin of j^{th} prosumer (P_j)

Output: Pay Matrix (PM)

Function `payMatrix`$(M,N,AM,bidC_i,bidP_j)$**:**

```
for i ← 1 to M do
    for j ← 1 to N do
        if i == M then
            PM_ij = AM_ij · (bidP_j + bidC_i)/2;
        else
            PM_ij = AM_ij · (bidP_j + bidC_{i+1})/2;
```

4. These first k prosumers are the winners of the double auction, and the group of these prosumers will act as our temporary leader for that time phase.

H. Super Leader Selection and Consensus Process

The "super leader" is the leader of all the temporary leaders. We have employed the PoS consensus algorithm for block creation among the temporary leaders. Temporary leaders can increase their chances of being selected as super leaders by putting in a large number of Pcoins. The temporary leaders put in a certain amount of Pcoins to indicate their interest in participating in the consensus process. Along with the amount of Pcoins, the concept of randomization is used to select the super leader. The probability of selection is influenced by the product of these two factors, which are the maximum Pcoins and the randomization. This means that temporary leaders with a large amount of Pcoins have a high chance of being selected as the super leader. By giving temporary leaders with large Pcoins a higher chance of being selected as super leaders, ETradeChain aims to encourage the nodes (prosumers) to generate the maximum possible green energy and avoid fossil fuel-based energy.

The Pcoins created in the last period of δT will be put at stake. The super leader gets remuneration for processing the transactions and creating the block. *Algorithm* 19 describes the PoS consensus for block creation and super leader selection. The algorithm for super leader selection and block creation works as follows:

1. All the nodes put their transactions in the transaction pool, which is accessible to all the other nodes. The temporary leaders verify the transactions in the pool.
2. The status of transactions in the pool is set to "pending" to indicate that they are waiting to be included in the next block creation. Once these transactions are included in the block, the status is updated to "processed."
3. All the temporary leaders put their recently generated Pcoins during δT time as a stake to participate in the super leader selection process. As per the algorithm, the super leader is selected randomly out of all the temporary leaders; however, those who put in the maximum Pcoins will have the highest chance of being selected as the super leader. The selected super leader is called the validator.

4. We have given priority to nodes who put maximum Pcoins at stake because it will motivate the nodes (prosumers) to generate the maximum possible green energy and avoid fossil fuel-based energy.

5. The validator starts forging the block. First, it collects the verified transactions from the transaction pool, creates the block by following the intermediate processes like Merkle roots (Mrt), previous block hash (PBH), and block hash (BH), and broadcasts the block in the network for verification.

6. After the verification, the newly created block is added to the existing blockchain, and the network is ready for the next round of blocks to be created, followed by the selection of the super leaders.

7. The other node in the network does the verification. If the other node finds any illegal activity by the leader, the first one to report it will receive the stake of the bad leader.

Algorithm 19: Super Leader Selection and Block Creation

```
Input: Pcoins ;                          /* Pcoins are taken as stake for consensus */
Output: Leader Node, Block (BL)
Function selectValidator(nodes, Pcoin, node.status):
    while node ≠ validator do
        stake ← putStake(Pcoins);
        selectValidator(stake+randomization and node.status);
Function blockGenerate(Tr_New):
    if validator.isForging then
        stake.Locked(node.stake);
        while TP ≠ empty or no_of_transaction = 516 do
            if TP.status == pending then
                Mrt ← createHashTree(TP.getAllTransactions);
                PBH ← H(PB) & BH = H(Mrt, T, PBH);
                block ← createBlock(Mrt, T, PBH, BH);
                broadcast_block(block);
                /* return to the initial step again to create a new block.     */
            else
                Waits for new transactions.;
Function blockVerification(block):
    while node.verifying do
        receive_block(block) & verify_block(block);
        if block == verified then
            stake.Unlocked(node.stake);
            node.status ← good;
            reward ← transaction_fees;
            TP.status ← processed;
        else
            stake.deduct(node.stake);
            node.status ← bad;
```

Remuneration Model

By compensating validators or super leaders for their efforts, block creation has been made more appealing to ETradeChain. The individual who contributes blocks to the chain receives a payment. Each block provided has a cost, which is referred to as transaction fees; hence, remuneration is decided by the blocks added by leaders. Therefore, every transaction is subject to a transaction fee. If every block contains N transactions, then the general equation for the total transaction fee (TTF) can be depicted as follows:

$$TTF = [\sum_{i=1}^{N} TF_i] \tag{5.36}$$

where TF is a transaction fee, and N is the number of transactions in some j^{th} block.

Avoid Malicious Block Generation

In the previous section, we described the remuneration model to reward the leader node for creating the block and appending it to the blockchain. However, applying this approach increases the direct motivation of the leader node to bunch the verified transactions and put them into the block. However, it reduces the motivation of other verifier consensus nodes. So, to solve this difficulty, ETradeChain employs a concept to encourage the verifier consensus nodes to participate honestly in the consensus process. Suppose any validator node has found that the newly created block is malicious or contains some malicious transactions (such as double-spent transactions). In that case, that node is awarded the total transaction fee TTF instead of the leader node, and the total Pcoins deposited by the leader node is also credited to that node. Additionally, the leader node is removed from the network as a punishment and marked as a malicious node. By applying this concept, all the miner nodes will actively participate in the consensus process, which will eliminate the possibility of various attacks.

5.3.5 Implementation Detail and Performance Analysis

In this section, we first give a detailed description of both schemes' implementation environments. Then, we show the performance analysis of both schemes. In SmartGrid-NG, we analyze the GridChain and the whole SmartGrid-NG framework separately. First, we describe the performance analysis of GridChain and explain the performance analysis of SmartGrid-NG. We evaluate and compare the performance of the GridChain based on the consensus algorithm and throughput and SmartGrid-NG with state-of-the-art schemes based on the two parameters, i.e., computation overhead and energy consumption. The performance of ETradeChain is measured based on the five parameters: consensus delay, throughout, computational cost, energy consumption, storage, and communication cost.

Implementation Detail of SmartGrid-NG

We have set up a testbed comprising a Computer System functioning as a Grid Operator and Raspberry Pi 4 model B (RP4) boards serving as consumers. The RP4 board embodies a device akin to a smart meter in its capabilities. We selected the RP4 to represent the smart meter and the integrated unit due to its ability to handle and store the security protocol. It comes equipped with a microcontroller for data processing and a storage unit for data retention—features that are essential for a smart meter. Additionally, this device can

establish communication with other devices via WiFi or Bluetooth. We utilized eight RP4 boards for our consumer setup, each with a 4GB main memory. The computer system operates on a Windows 11 platform and boasts hardware specifications like a 4.4 GHz Intel 64-bit Core i5 processor and 8 GB DDR4 RAM. On the software front, it's equipped with Netbeans 8.2 and Java 8. As for the RP4s, they run on the Raspberry Pi OS (formerly known as Raspbian OS), a Debian-based OS tailored for the RP4. Each RP4 is powered by a 1.5 GHz Broadcom BCM2711 quad-core cortex-A72 processor. We employed the Java programming language to create programs for the testbed, encompassing one grid operator and eight consumers. Figure 5.19 illustrate the testbed setup of SmartGrid-NG.

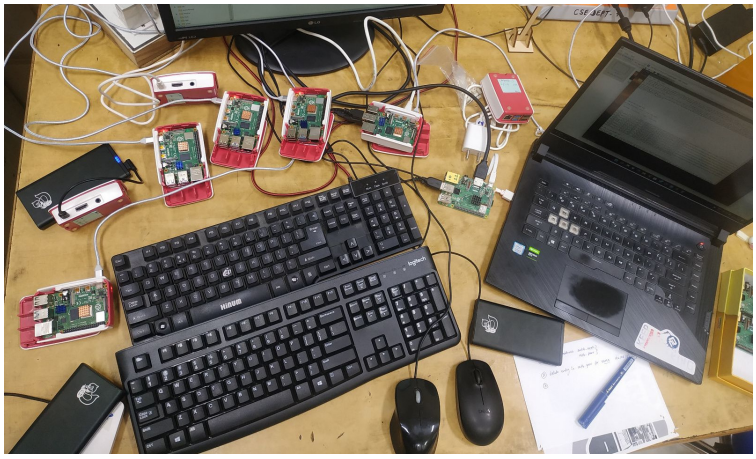


Figure 5.19: Testbed setup for SmartGrid-NG using one computer system and eight Raspberry Pi boards

Implementation Detail of ETradechain

We have developed a testbed that consists of 32 Raspberry Pi 4 model B (RP4) boards. Out of 32, 1 is represented as a smart grid (shown in the system model). 8 out of 32 RP4 boards act as prosumer nodes, and the rest of the RP4 boards act as consumers, where each prosumer is connected to 3 RP4 board consumers. The consumer RP4 board represents a device with the capability of a smart meter; the prosumer RP4 board acts as a smart meter and integrated unit. We chose the RP4 as the smart meter and integrated unit because it has the capacity to process and store the security protocol. It has a microcontroller for data processing and a memory unit for data storage. The smart meter should also have these properties. The device can also communicate with other devices over a WiFi or Bluetooth wireless connection. We have used two types of RP4 boards with different configurations. The RP4 for consumers has 2 GB of memory and 4GB for prosumers. Both boards have the same software and hardware configurations except for the memory. The operating system installed in each RP4 is Raspberry Pi OS, also called Raspbian OS, a Debian-based operating system for RP4. Each RP4 is equipped with a 1.5 GHz Broadcom BCM2711 quad-core cortex-A72 processor. We have used the Java programming language to develop the programs for the testbed. Figure 5.20 shows the implementation environment of ETradeChain. This figure shows seven prosumer RPs, eight consumer RPs, and one smart grid RP4 unit. We show only a few RP4s in our testbed image for clear visibility. Figure 5.21 shows the internal working process of the proposed scheme in the RP4 module. Figure 5.21a shows that the prosumer generates the block. Figure 5.21b shows the consensus process is taking place, i.e., the selection of a super leader.

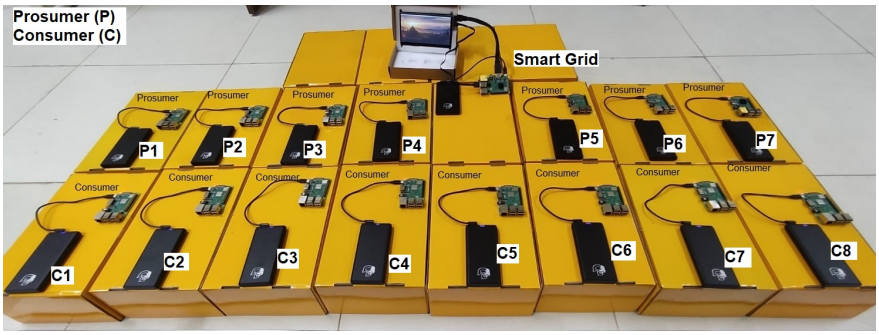


Figure 5.20: Implementation environment for ETradeChain testbed

Evaluation of GridChain

The GridChain consensus algorithm is evaluated based on consensus delay and throughput parameters. **Consensus Delay:** The consensus delay of GridChain is better than state-of-the-art schemes for two reasons.

- Leader-Based Consensus: GridChain's leader-based consensus mechanism ensures that only one node (the leader) is responsible for block generation at any given time. This eliminates the need for multiple nodes to compete for block generation, as seen in state-of-the-art schemes like PoW. This focused approach can significantly reduce the time taken to reach a consensus.
- Reputation-Based Leader Selection: Another reason is that, by selecting leader nodes based on reputation, GridChain ensures that only the most trustworthy and efficient nodes take the lead. This can reduce the chances of selecting a malicious or slow node as a leader, which could otherwise introduce delays in the consensus process. However, some of the other schemes employ consensus mechanisms that require nodes to solve cryptographic puzzles or engage in multiple rounds of communication. Such processes can be time-consuming, leading to higher consensus delays.

Throughput: Like the consensus delay, the throughput of the GridChain is also better compared with state-of-the-art schemes. Below are the three reasons to support the better performance of the GridChain.

- Efficient Block Verification: In GridChain's block verification process, the leader node sends the new block with a nonce value, and verification nodes recreate the block to check its validity. This direct approach can lead to faster block verifications, allowing the system to process more blocks in a given timeframe, thereby increasing throughput.
- Reputation System: By continuously evaluating the behaviour of consensus nodes, GridChain ensures that only efficient and trustworthy nodes participate actively. This can lead to faster transaction verifications and block additions, enhancing the system's overall throughput.
- Intricate Verification Processes: Some of the other schemes have more intricate block verification processes, which could involve multiple rounds of communication or complex cryptographic verifications. Such processes can be time-consuming, limiting the number of transactions and blocks that can be processed in a given time frame and reducing throughput.

```
pi@raspberrypt :~ /Desktop/EnTrade/Block/Client

a876e4628f50f7e721dffac18e1abb2a5f06bef02cac4\",\n \"TransID\": 500,1
n \"CoinID\": 500,\n \"CoinStatus\": \"U\", \n \"BlockID\": 500, \n
\"BlockTransactionNo\": 1,\n \"Sender\": \"USER11\", \n \"Receiver\":
\"SMARTGRID\",\n \"TransactionType\": \"C\", \n \"TransactionDate\":
\"2021-11-12 00:00:00\", \n \"Broadcasted\": \"Y\", \n \"previousHash
\": \"39a9ed60769f055904eba8fb9fe86a239b091af6bbb2c6116754a4bae34ee73a
\",\n \"timeStamp\": 0, \n \"nonce\": 0\n}", "ReceiverPort": "9004", "Re
ceiverUsername":"SMARTGRID", "Sender IPAddress":"192. 168.137. 21",
"hash":
"f5a4c2055670ffb617ea876e4628f50f7e721dffac18e1abb2a5f06bef02cac4","Re
ceiverIPAddress":"192.168.137.10"}ENTRADEM20CS017MIGIAkIAzajsnsocpj4Nw
idQtUrNbCgIkglgY0TckmbogUiv9y04y7DZAZVCXHCHLGYOHRxjQOWYHtaeE5VqgV1RRmF
OV2WCQgHk6410ykHJwy4EeGBepJogIh5Ri4E4V5AamYGoBjZUR+lYwBjNF6ptORh7VqN8s
AB8I8NtDnZ6bjYlQfuSeb5Zog ==
i am finally done
PBFT Time2631
POS Time896
Block Creation Time: 384670
pi@raspberrypi :- /Desktop/EnTrade/Block/client $
```

(a) Prosumer Generating Block

```
pi@raspberrypi: ~/Desktop/EnTrade/Block/Server

Valid user, processing ahead
POSFLAG
inside stringtosecretkey:e40yBMNTfMSgMgwhOa301EY56FVfLi/7ewtgs5GPt48=
Valid user, processing ahead
Received Super Leader
inside stringtoSecretkey:e40yBMNTfMSgMgwhOa30LEY56FVfLi/7eWtgs5GPt48=
Valid user, processing ahead
POSFLAG
inside stringtosecretkey:e40yBMNTfMSgMgwhOa301EY56FVfLi/7ewtgs5GPt48=
valid user, processing ahead
POSFLAG
inside stringtoSecretKey:e40yBMNTfMSgMgwh0a301EY56FVfLi/7eWtgs5GPt48=
Valid user, processing ahead
Received Super Leader
inside stringtoSecretKey:e40yBMNTfMSgMgwhOa301EY56FVfLi/7ewtgs5GPt48=
Valid user, processing ahead
Received Super Leader
```

(b) Consensus Algorithm Processing

Figure 5.21: ETradeChain working process on the RP4 devices. These are the command line interface of RP4 doing all the required processing

The evaluation results show that GridChain performed very well compared with state-of-the-art schemes while maintaining a high level of security. Most of the previous cryptocurrencies have fork generation issues that are addressed by the proposed GridChain protocol. To reduce the propagation delay, we have fixed the upper bound of the block size to 1 MB. The maximum number of transactions in a single block is 516 in order to keep the block size under 1 MB. Figure 5.22a shows the improved performance of GridChain for consensus delay compared with state-of-the-art schemes. In GridChain, the consensus delay is reduced by approximately 80% compared with the other schemes. Figure 5.22b depicts the GridChain's throughput. The results show that GridChain outperformed other blockchain protocols in terms of performance. GridChain-NG's throughput has also been tested with varying time intervals between the leader nodes, as presented below.

Evaluation of SmartGrid-NG

We evaluate and compare the performance of the SmartGrid-NG with state-of-the-art schemes based on the mentioned two parameters, i.e., computation overhead and energy consumption. We have compared

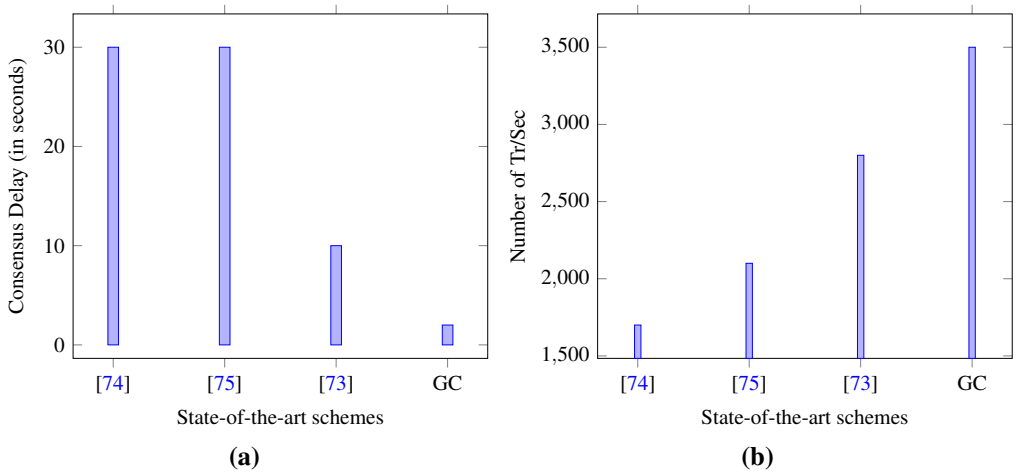


Figure 5.22: Comparative analysis of GridChain based on (a) consensus delay and (b) throughput (number of transactions executed per second) with state-of-the-art schemes

the performance of SmartGrid-NG with state-of-the-art schemes [122], [169], [119], [170], and [118]. We selected these schemes from the others mentioned in the related work section for the following reasons: The selected papers align more closely with the proposed SmartGrid-NG. Additionally, these selected papers provide data and results that aided our understanding of the schemes and facilitated comparison analysis. Computational cost is determined by the time taken to execute various cryptographic tasks within the framework. These tasks include authentication, verification, communication, and the creation and verification of blocks. The average execution time (after performing the execution 1000 times) for the different cryptographic functions is shown in Table 5.13. This table also illustrates the time required to execute these cryptographic functions on the specified platforms, namely the computer system and Raspberry Pi 4 model. A comparison of the computational cost of SmartGrid-NG with state-of-the-art is presented in Table 5.14. Once the computational cost is determined, it's possible to compute the energy usage using the formula, $E_{rg} = T_{ct} * C_p$. In our testing environment, the desktop system consumes 10.88 W of power for computing one task, while the Raspberry Pi 4 consumes 2.4W. As shown in Table 5.14, it is evident that the computation cost of SmartGrid-NG is significantly lower compared to state-of-the-art schemes. This is attributed to the number of cryptographic functions used throughout the process, from generating transactions to their verification and addition to the blockchain ledger. The state-of-the-art schemes employed intensive cryptographic functions, which increased the computational cost. Consequently, the energy consumption at the device level for executing operations is also high. The last two columns of Table 5.14 clearly highlight the lower energy consumption of SmartGrid-NG compared to state-of-the-art schemes.

Table 5.13: Individual computation time of different cryptographic functions for smart grid testbed

Functions	**On Desktop**	**On Raspberry Pi 4GB**	**On Raspberry Pi 2GB**
SHA256 Hash function: $T_{H()}$	0.0008	0.005	0.045
ECC Crypto-system: T_{ECC}	0.053	1.6	3.9
ECDSA signature: T_{sign}	0.071	2.94	6.13
ECDSA verification: T_{veri}	0.065	2.86	5.63
point multiplication: $T_{mul(.)}$	0.117	5.0	11.0
AES Crypto-system 192-bit: T_{AES}	0.125	5.96	12.71
Modulo operation: $T_{mod(.)}$	0.067	2.0	5.0

Table 5.14: Performance analysis of SmartGrid-NG based on the comparison of computation cost (CC) in milliseconds and energy consumption (EC) in Joule on Desktop and Raspberry Pi 4 2GB (RP) with state-of-the-art schemes

Scheme	Computation Cost Equation	CC on Desktop	CC on RP4	EC on Desktop	EC on RP4
[122]	$5T_{H(.)}+3T_{sign}+3T_{veri}+3T_{ECC}$	0.592	22.23	178.704	85.068
[119]	$6T_{H(.)}+4T_{sign}+4T_{veri}$	0.549	23.79	126.53	57.096
[169]	$6T_{H(.)}+3T_{sign}+3T_{veri}$	0.413	17.43	189.639	85.32
[170]	$6T_{AES}+2T_{sign}+2T_{veri}$	1.02	47.36	515.277	167.472
[118]	$5T_{sign}+5T_{veri}+2T_{ECC}$	0.786	32.20	350.336	159.84
S-NG	$3T_{H(.)}+3T_{sign}+3T_{veri}$	0.410	17.41	63.267	28.548

Evaluation of ETradeChain

We have evaluated ETradeChain based on six parameters, which are consensus delay, throughput, computation cost, energy consumption, storage cost, and communication cost.

1. Consensus Delay and Throughput: When evaluating ETradeChain, we look at the number of transactions per second, the time it takes to reach an agreement, and how fair it is when selecting new leaders. The assessment results demonstrate that, without compromising the security level, reducing consensus latency and increasing throughput is feasible by altering the configuration of the miner nodes, i.e., two-level leader selection, first auction-based temporary leader selection, and a second super leader. Nearly every statistic across all of the tested characteristics shows that ETradeChain is superior to the competition. The state-of-the-art blockchain methods have a common weakness: fork creation and high consensus delay. ETradeChain addressed these weaknesses. To lower the propagation delay in the blockchain network, we fix the block size at less than or equal to 1 MB. Each block contains a maximum of 516 transactions, making the block size less than 1 MB. Each block contains three components: the block header, transactions, and transaction count. The block header size is 80 bytes, the transaction count is 4 bytes, and the remaining space, i.e., 10,48,492 bytes, is occupied by the transactions. Each transaction has a maximum of 2000 bytes. Therefore, each block can contain a maximum of 516 transactions. Figure 5.23a shows the consensus delay of ETradeChain compared with state-of-the-art schemes. The result shows that the consensus delay of ETradeChain is approximately 90% low as compared to other blockchain protocols.

Figure 5.23b depicts ETradeChain's throughput. When compared to other schemes, ETradeChain performed exceptionally well. Changing the time gap between leader nodes changed the throughput of the ETradeChain. As the number of forgers grows, the likelihood that various forgers will have an equal probability of generating a block increases. The ETradeChain is fairer than other platforms. The state-of-the-art schemes show significant congestion in the network and unnecessary forks due to the competition in solving computational puzzles first to win the reward. In ETradeChain, the architecture first chooses the validator as a temporary leader and then as a super leader. This reduces congestion and increases throughput.

2. Computational Cost and Energy Consumption: Computational time is the time taken by the Testbed for the execution. The measuring unit is milliseconds and is represented as T_{ct}. We illustrate the computational cost (also known as execution time) based on the operational costs for authentication, verification, communication, and block creation and verification. Table 5.13 shows different cryptographic functions and their computation time on both types of RP4 boards. The time taken by all these cryptographic operations is based on the implementation environment we defined earlier. Table 5.15 shows the comparative computation cost of ETradeChain versus state-of-the-art schemes. The energy consumption of a system is the amount of energy consumed by the system in order to carry out its operations of the system. During

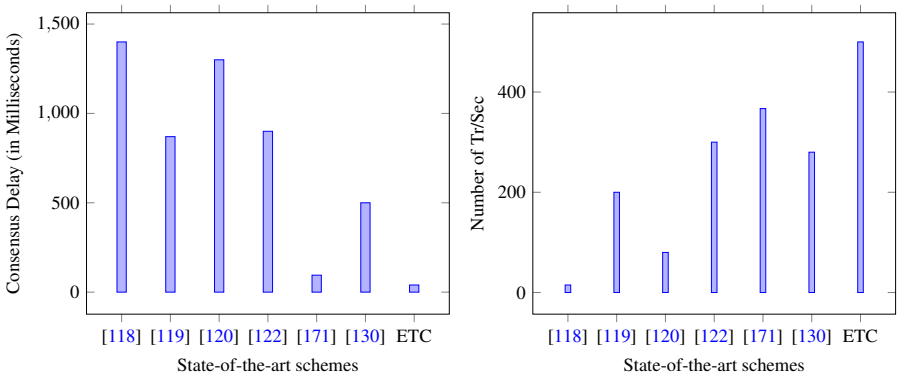


Figure 5.23: Comparative analysis of ETradeChain based on (a) consensus delay and (b) throughput with state-of-the-art schemes

the execution process, the system requires a certain amount of energy in order to complete all of the phases of the scheme. The equation $E_{rg} = T_{ct} * C_P$ is used to calculate energy consumption, where C_P is the CPU power (5.6 W for RP4 4GB and 2.4 W for RP4 2 GB) and T_{ct} is the computational cost. The computation cost was already measured in the previous section. Like computation cost, energy consumption is also measured on two platforms, i.e., RP4 with 4 GB and RP4 with 2 GB. Table 5.15 shows the comparison of ETradeChain with state-of-the-art schemes.

Table 5.15: Performance analysis of ETradeChain based on the comparison of computation cost (CC) in milliseconds and energy consumption (EC) in Joule of ETradeChain on Raspberry Pi 4 (RP) 2GB and 4GB with state-of-the-art schemes

Scheme	Computation Cost Equation	CC on RP4 4GB	CC on RP4 2GB	EC on RP4 4GB	EC on RP4 2GB
[170]	$6T_{AES} + 2T_{sign} + 2T_{veri}$	47.36	99.78	265.22	239.48
[118]	$5T_{sign} + 5T_{veri}$	29	59.95	162.4	143.88
[119]	$6T_{H(.)} + 2T_{sign} + 2T_{veri}$	11.63	24.25	65.128	58.20
[120]	$11T_{H(.)} + 8T_{mul(.)}$	40.05	88.50	224.28	212.38
[122]	$5T_{H(.)} + 3T_{sign} + 3T_{veri}$	17.43	35.97	97.61	86.33
ETradeChain	$3T_{H(.)} + 1T_{sign} + 1T_{veri}$	5.82	12.13	32.54	29.11

3. Storage and Communication Cost: There are various computations performed in the processing of transactions and blockchain operations, such as computing hash code, secret keys, digital signatures and many others. These computations consume some memory space on the node side and in communications. Therefore, the storage cost refers to the storage space required for storing the output value of the computing operations. The measuring unit for storage cost is bytes and represented as S_{sc}. Generally, a Key S_k requires 24B (B=Bytes), ECC encryption/decryption S_{ECC} takes 48B, AES encryption/decryption S_{AES} takes 24B, a hash function (SHA-256 bit) $S_{H(.)}$ takes 32B, and a signature S_{sign} and verification S_{veri} requires 10B. Table 5.16 shows the comparative cost of ETradeChain along with the state-of-the-art schemes. As shown in Table, 5.16, the number of operations in ETradeChain is much less compared with state-of-the-art schemes. Therefore, the storage cost is also very low in ETradeChain.

Communication cost refers to the storage required for storing the communicating variables such as hash code, ciphertext, and other parameters. The measuring unit is a byte and is represented as C_{cc}. The operations and variables that have been used in communication between the smart meter and the miner

node are the hash function and the ECDSA digital signature. Table 5.16 shows the comparative analysis of different mechanisms against ETradeChain.

Table 5.16: Comparison of storage and communication cost of ETradeChain with state-of-the-art schemes

Scheme	Storage Cost	Cost in MB	Communication Cost	Cost in MB
[170]	$2S_{AES}+2S_{sign}$	52	$4S_{AES}+2S_{sign}$	84
[118]	$8S_{H(.)}+8S_{sign}$	592	$8S_{H(.)}+8S_{sign}$	592
[119]	$5S_{H(.)}+2S_{sign}$	340	$3S_{H(.)}+1S_{sign}$	202
[120]	$10S_{H(.)}+3S_{mul(.)}$	700	$ElementData(230Bytes)$	230
[122]	$3S_{H(.)}+1S_{sign}$	202	$3S_{H(.)}+1S_{sign}$	202
ETradeChain	$2S_{H(.)}+1S_{sign}$	74	$2S_{H(.)}+3S_{sign}$	94

5.3.6 Security Analysis for SmartGrid-NG and ETradeChain

In this section, we have shown the security analysis of the proposed GridChain and SmartGrid-NG. We have shown that the GridChain is secure from double-spending attacks and fork attacks. Also, we have shown that the SmartGrid-NG achieved all the security requirements mentioned in our contribution, which are transaction integrity, confidentiality, authentication, and fairness. In this section, we have also explained the auction property analysis and security analysis of ETradeChain. The security analysis of the ETradeChain justifies the prevention of double-spending attacks, the prevention of forking, fault tolerance, and non-repudiation. Below, we have proven with the help of a lemma and proof that the system is protected against the attacks mentioned above. Table 5.17 shows the security comparison of ETradeChain with the state-of-the-art approach where $\surd$ represents "Secure from Attack" and $\otimes$ illustrates "Not Secure from Attack."

Table 5.17: Security comparisons of SmartGrid-NG and ETradeChain with state-of-the-art schemes

Attacks	[118]	[119]	[120]	[122]	[171]	[130]	SmartGrid-NG	*ETradeChain*
Double-Spending	$\surd$	$\surd$	$\otimes$	$\surd$	$\otimes$	$\otimes$	$\surd$	$\surd$
Forking	$\otimes$	$\otimes$	$\surd$	$\surd$	$\otimes$	$\surd$	$\surd$	$\surd$
Non-Repudiation	$\otimes$	$\otimes$	$\surd$	$\otimes$	$\surd$	$\otimes$	$\surd$	$\surd$

Lemma 14. *The risk of a double spending attack is minimized in GridChain based on the attacker's computational power (q) in comparison with honest node (p). Thus, the GridChain protocol prevents the double-spending attack.*

Proof. Following the binomial random walk [172], a race condition can be modelled. A race condition R is the difference in the computational capacity of mining blocks between honest (p) and malicious $(q = 1-p)$ nodes, i.e., the attack can be derived as

$$R_{i+1} = \begin{cases} R_i + 1, for\ p \\ R_i - 1, for\ q \end{cases} \tag{5.37}$$

where i is the individual block generation race. It is clear from (5.37), that if $q > p$, then the attacker continues to generate the blocks until $R < 0$, meaning it can replace all the blocks generated by the honest

node. Therefore, the probability of an attacker overtaking the honest node using Gambler's Ruin Problem [173] can be modelled as

$$Pr_i = qPr_{i+1} + pPr_{i-1} \tag{5.38}$$

since, p+q=1, the equation (5.38) can be rewritten as

$$Pr_{i+1} - Pr_i = \frac{p}{q}(Pr_i - Pr_{i-1}) \tag{5.39}$$

The equation (5.39) can written in generalized form as

$$Pr_{i+1} = Pr_1 \sum_{j=0}^{i} (\frac{p}{q})^j = \begin{cases} Pr_1 \frac{(1-\frac{p}{q})^{i+1}}{1-\frac{p}{q}}, if\ p \neq q \\ Pr_1(i+1), if\ p = q = 0.5 \end{cases} \tag{5.40}$$

Let, $i = N-1, Pr_N = 1$ By solving Pr_1, we will get,

$$Pr_i = \begin{cases} \frac{1-(\frac{p}{q})^i}{1-(\frac{p}{q})^N}, if\ p \neq q \\ \frac{i}{N}, if\ p = q = 0.5 \end{cases} \tag{5.41}$$

If we consider that the attacker has unlimited computational power, i.e., $i \longrightarrow \infty$, and $q > p$, then,

$$\lim_{i \to \infty} \frac{1-(\frac{p}{q})^i}{1-(\frac{p}{q})^N} = 1 \tag{5.42}$$

By normalizing the (5.42), we will get the probability of the attacker overtaking the honest node, which can be defined as

$$Qr_z = \begin{cases} (\frac{q}{p})^{z+1}, if\ q < p, z \geq 0 \\ 1, if\ q > p, z < 0 \end{cases} \tag{5.43}$$

It is clear from the above mathematical derivation that, in the proposed permissioned blockchain protocol GridChain, if the attacker's computational power (q) of block generation is higher than the computational power of the honest node (p), then there is a possibility of a double-spending attack. However, GridChain is a leader-based consensus algorithm where the leader is responsible for generating blocks, and the rest of the network is responsible for block verification. Therefore, to launch a successful attack, an attacker needs to increase the computational power greater than the whole network, which is nearly impossible. Thus, GridChain is protected from double-spending attacks. □

Lemma 15. *Let h_b be the height of honest block b and $h_{b'}$ is the height of malicious block, where $h_b = h_{b'}$, then a fork will occur if $b, b' \in \mathbb{B}$ and $b \neq b'$. The GridChain prevents the fork because if $b \neq b' \in \mathbb{B}$, then $h_b \neq h_{b'}$.*

Proof. As per the statement in the lemma, a fork condition can also be illustrated in the following form:

$$\exists b, b' \in \mathbb{B}\ and\ b \neq b' | h_b = h_{b'} \tag{5.44}$$

Consider a node in $\mathbb{B}$ of height h_b gets a block b' with a height of $h_{b'}$ such that $h_{b'} > h_b$. According to the longest chain rule, the new height of blockchain $\mathbb{B}$ changed to $h_{b'}$. Also, block b is the ancestor of block b'. However, if both the blocks are added in different branches, the block b will be orphaned, i.e., removed

from the main branch. Further, the local state of the $\mathbb{B}$ maintained by the node should be consistent with the rest of the network, considering this as the longest chain. GridChain follows the leader-based consensus algorithm, where only one node is responsible for generating the block at a time. Suppose two different blocks arrive at the same time. In that case, it is clear that one of the blocks is malicious, which will be identified by the network and later discarded. Therefore, two different blocks b and b' can not belong to the same blockchain $\mathbb{B}$ at the same height, i.e., $h_b \neq h_{b'}$. □

Lemma 16. *Let Tr be the transaction generated by smart meter SM_i, $i = 1,2,...,N$, and Λ is the adversary who might attack and alter its integrity of it. The SmartGrid-NG provides the authentication and integrity of data shared using MAC and a secure hash function.*

Proof. The proposed SmartGrid-NG is secured from transaction alteration and authentication, which can be illustrated as follows.

1. To secure the integrity of transactions, the proposed SmartGrid-NG does the, the smart meter SM_i, $i = 1,2,...,N$ and consensus node CN_j, $j = 1,2,...,M$ compute the $MAC_{Sk_i^{CH_j}}$ and $MAC_{CH_j{}^{Sk_i}}$, where MAC_{sk} is message authentication technique utilizing the one-way secure hash function $SHA256$ with the secret key sk.
2. It is worth noting that, the security of MAC_{sk} depneds on the security of sk and the hash function used which is $SHA256$ in our case. Due to the discrete logarithm problem (DLP) on the group G (generator of ECC), the adversary Λ can not obtain the shared secret session key $sk = Sk_i^{CN_j} = Sk_{CN_j}^i$. Therefore, Λ can not forge and tamper transactions or unique identities of SM_i and the consensus node CN_j.
3. Besides, the transaction and data sharing between SM_i and CN_j incorporate the timestamp t_τ in computing the shared transactions and MAC_{sk}. Because only the MAC_{sk} with current timestamp t_τ can go through the successful verification process. In this way, the SmartGrid-NG is also prevented from message replay attacks.
4. Thus, it is ensured that, in SmartGrid-NG, the data is authenticated at the receiving end, and its integrity is ensured. Let's consider that an adversary is able to get transactions while in transit and tries to do some modification. To resend these transactions, the signature of a legitimate sender is required, which is not possible, as we have assumed that the adversary can not get the private key of any node. Thus, the scheme ensures the integrity of the transactions.

□

Lemma 17. *Let Tr be the transaction generated by smart meter SM_i, $i = 1,2,...,N$, and Λ is the adversary who might attack and expose the confidentiality of the communication. The SmartGrid-NG provides confidentiality using the Elliptic curve Diffie–Hellman (ECDH) and secure hash function.*

Proof. The SmartGrid-NG is secured from any type of confidentiality attack due to the following computation in the communication between SM_i and CN_j and also between CN_i and CN_j.

1. According to the adversary model defined above, it is ensured that the Λ can not calculate the $Sk_i^{CN_j} = Pr_{SM_i} * Pb_{CN_j}$ and $Sk_{CN_j}^i = Pr_{CN_j} * Pb_{SM_i}$ due to the computation diffie-hellman problem.

2. Due to the shared session key being calculated at the SM_i end and CN_j end, it is impossible for Λ to obtain the shared secret session key. Therefore, Λ can not decrypt the CT_{Tr_k} without knowing the secret key. Thus, confidentiality is ensured.

□

Lemma 18. *The GridChain employed the reputation value for fair leader selection for the consensus mechanism. Thus it provides fairness in leader selection and is free from unbiased consensus.*

Proof. The leader node is responsible for every new block generation and verification of it. Therefore, a fair leader node selection algorithm is crucial for the unbiased behaviour of the method. The fairness of leader selection in GridChain can be expressed as follows to show the unbiased nature of it.

1. We have computed the reputation value $Rep_{CN_j}^{final,k}$ using the reputation updation model based on opinion calculation and interaction behavior.
2. Node reputation updation model is based on the many parameters and recent and past behavior of the node, such as positive and negative behavior, and effectiveness of the recent interactions. Through the reputation computation model, we calculate the trust ($T_{i,j}$) and uncertainty ($U_{i,j}$) as mentioned in subsection 4.4.3. Also, the final reputation value is computed by the current leader node CN_x, which is $Rep_{CN_j}^{(final,k)}$ in the k^{th} round of the consensus.
3. It is ensured that the node with the highest reputation value will not always be selected because, in the leader selection section, we have defined that in every round, a new group is formed based on the threshold value, and out of them, one node is selected. Thus, GridChain ensured fairness in the leader selection and consensus mechanism.

□

Lemma 19. *ETradeChain's auction is budget balanced.*

Proof. From the Algorithm 18, we can see that the total payment received by the j^{th} prosumer is the sum of the j^{th} column in the pay matrix. Every column contains i rows. Therefore, the total payment received by j^{th} prosumer is $Pay_j = \sum_{i=1}^{M} PM_{ij}$. Similarly, total payment given by i^{th} consumer will be $Pay_i = \sum_{j=1}^{N} PM_{ij}$. The total payment received by all the prosumers will be

$$TotalPay_j = \sum_{j=1}^{N}\sum_{i=1}^{M} PM_{ji} \tag{5.45}$$

Similarly, the total payment given by all the consumers can be represented as,

$$TotalPay_i = \sum_{i=1}^{M}\sum_{j=1}^{N} PM_{ij} \tag{5.46}$$

From equation (5.45) and (5.46), we can see that the Total payment received by prosumers is equal to the total payment given by the consumers. Therefore, we proved that our system is budget-balanced. □

Lemma 20. *ETradeChain's auction satisfies individual rationality.*

Proof. Individual Rationality means no peer should lose by joining the auction. In our case, the utility of consumers will never be negative. As per the condition in Algorithm 17, we are ensuring that in the allocation scheme, the bid price of the consumer should be greater than or equal to that of the prosumer. Also, considering the equation (5.35), we can say that the buyer will never pay more than the bid value, and the seller will never receive less than the asking value. Therefore, the utility of the buyer and seller will never be negative. It will be zero if they lose. Therefore, individual rationality is satisfied. □

Lemma 21. *The ETradeChain protocol ensures no fork generation in the blockchain or that there is no possibility of parallel block generation with the same set of transactions.*

Proof. The fork generation can be easily understood. The minimum difference between the number of blocks generated by a malicious node and a fair node to confirm a transaction and lower the blockchain network's vulnerability probability than the threshold $\varepsilon \in (0,1]$ is defined as follows:

$$\alpha^*(\varepsilon) = min\{\alpha : P(Q > \alpha) \leq \varepsilon\} \tag{5.47}$$

$$\alpha^* = -\log(\varepsilon)/\log(\lambda/\Lambda) \tag{5.48}$$

where P is the probability of a malicious node generating the block before the fair node. The high value of α^* has the efficiency to counterattack the intentional forks. To ensure blockchain network stability, fair node computing power must be increased. The computing power required to finish the forging in the allotted time frame is equal to the fair node's effective computational power. ETC's fair leader node has a lot more computing power than the malicious node in order to generate a block. As a result, an attacker is unable to fork the ETradeChain. ETradeChain also does not enable many nodes to forge the same block simultaneously, meaning that only the fair super leader node can forge the block at any one time. Thus, fork formation in the ETradeChain is less likely to occur.

However, ETradeChain has a time period in which the leader creates the blocks. This is in contrast to other blockchain protocols like Bitcoin-NG [70], and OmniLedger [74] that have fork generation problems. After the interval has passed, the next node in the network generates the block after selecting it as the super leader node. Due to this, ETradeChain does not have the problem of fork formation because two forger nodes cannot generate the block simultaneously. As a result, ETradeChain has no issues with fork creation. □

5.4 PRIVACY-PRESERVING ENERGY TRADING IN SMART GRID

With the inclusion of blockchain technology in smart energy, consumers can also harvest solar energy and sell the surplus energy to neighbours or the main grid. Such consumers are called prosumers. This advancement solved much of the problem of long-distance energy transmission. Additionally, this is also cost-effective for consumers [174]. In transaction processing, sensitive information is included, which needs

to be protected. Therefore, user data privacy is a big concern [49]. Additionally, the existing blockchain solutions are not suitable for applications like a smart grid because they do not offer any privacy preservation of sensitive information [175]. However, privacy is achieved in the blockchain-based cryptocurrency Zcash using the zk-SNARK concept in [176]. Similarly, privacy in electronic devices in consortium blockchain is achieved in [177].

5.4.1 Techniques Used to Provide Privacy in Blockchain

Following are the widely used techniques for privacy preservation.

1. **Zero-Knowledge Proofs (ZKPs):** A cryptographic method where one party can prove to another that a statement is true without revealing any specific details about the statement itself. Cryptocurrencies like Zcash [31] employ ZKPs (specifically zk-SNARKs [32]) to ensure transactions are recorded, but details like sender, receiver, and transaction amount remain concealed [33].

2. **Traceable Ring Signatures:** This technique, integral to the Monero cryptocurrency, ensures sender anonymity by allowing a transaction to be signed and verified by a group. Within this group, the actual signer remains anonymous, ensuring transactional privacy [178].

3. **Homomorphic Encryption:** A groundbreaking technique, homomorphic encryption allows computations on encrypted data (ciphertexts), producing an encrypted result. When decrypted, this result matches as if operations were performed on the original data, ensuring data privacy even during computations [179].

4. **Private Sidechains:** These are auxiliary blockchains running parallel to the primary chain, facilitating private transactions. Once a transaction concludes, only its final state might be recorded on the main chain, keeping transactional nuances private [97].

5.4.2 *zk-SNARK* Background

A *zk-SNARK* is a widely used scheme for zero-knowledge proof, which enables the provers to prove the validity of a given assertion to any verifier without revealing any information [180], [181]. It can be represented as the tuple of probabilistic polynomial algorithms $\zeta_{zk} = (Preprocessing, Prove, Verify)$ such that:

1. *Preprocessing* ($S(\mathbb{C})$): Given a arithmetic circuit $\mathbb{C}$, the preprocessing $S(\mathbb{C}$ produces two public parameters, the S_P and S_V for the prover and verifier respectively. It is important to note that the further process is dependent on these parameters.

$$S(\mathbb{C}) \longrightarrow (S_P, S_V)$$

2. *Prove* $P()$: It is the algorithm that is responsible for generating the proof π provided the public statement X and secret statement ω in $\mathbb{C}$.

$$P(S_P, x, \omega) \longrightarrow \pi$$

3. *Verifier* $V()$: The verification algorithm uses the public statement x and the proof π to verifiy the secret witness ω.

$$V(S_V, x, \pi) \longrightarrow \begin{cases} 0; Accept \\ 1; Reject \end{cases}$$

It is important to note that the verification process reveals nothing about the witness or secret value ω.

Definition 6. *Let $\mathbb{C}$ be an arithmetic circuit, P is the prover, V is the verifier, π is the proof, and ω is the witness, then $(\mathbb{C}, P, V)$ is zero knowledge if for every statement $x \in \mathbb{F}^n$, the proof π generated by prover reveals nothing about the ω. The Proof π reveals nothing about ω if the verifier can generate π on its own and learnt nothing from the π.*

5.4.3 Public Traceability

A traceable ring signature scheme requires that the following condition must hold: For every $k \in N$, every $n \in N$, every i,$i \in N = 1, ..., n$, every $issue \in 0, 1^*$, and every m, $m \in 0, 1^*$, if $(pk_N, sk_N) \leftarrow Gen(1^k)$, $\sigma \leftarrow Sig_{ski}(L, m)$, where $L = (issue, pk_N)$, and $\sigma \leftarrow Sig_{ski}(L, m)$, it holds with an overwhelming probability $(\in k)$ that

$$T(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg}) = \begin{cases} indep & if\ N_i \neq N'_i, N_i \in N \\ linked & else\ if\ Msg = Msg' \\ Pubk_{N_i} & otherwise \end{cases} \tag{5.49}$$

The equation 5.49 can be interpreted as follows. The function $\mathbb{T}()$ never outputs "linked" if $Msg \neq MSg'$. If a traceable signature scheme has tag-linkability (along with public traceability), $\mathbb{T}(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg})$ = "indep" implies that these two signatures are generated by different signers.

Third things is that $\mathbb{T}(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg}) = Pubk_{N_i}$ implies that they are signed by the same signer N_i. In addition, if $Msg \neq MSg'$, Trace never outputs "linked." Public traceability is a correctness condition; that is, it does not assure that the opposite holds. However, if a traceable signature scheme has tag-linkability (along with public traceability), $Trace(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg})$ = "indep" implies that these two signatures are generated by different signers. If it has exculpability,

$$Trace(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg}) = pk_{N_i}$$

implies that they are signed by the same signer i. Note that $Trace(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg})$ = "linked" doesn't mean that they are always generated by the same signer (because anyone can make a "dead" copy of any signature).

5.4.4 *zk-SGB*: Privacy-Preserved Blockchain Framework for Energy Trading in Smart Grid Using Zero Knowledge Proof

We design and develop a privacy-preserved energy trading system called *zk-SGB* using blockchain technology, which is more reliable, scalable, robust, and resistant to cyber-attacks. For the best result possible in *zk-SGB* scheme, we have also proposed a novel consensus algorithm called Weighted Node Selection Byzantine Fault Tolerance (WNS-BFT), which is suitable for permissioned blockchain Smart Grid network.

It provides a fair and robust consensus scheme utilizing active, honest past participation and computational power as the core element to select the honest group of consensus nodes. Additionally, *zk-SGB* integrated with the Zero-Knowledge Succinct Non-interactive Argument of Knowledge (zk-SNARK) to preserve privacy. It has three components, i.e., Smart Grid (Energy Provider), Prosumer (Energy Seller + Buyer), and Consumer (Buyer). All the components are connected, and relevant information is exchanged. The smart meters are installed at the consumer's place and monitor the consumer's power consumption. The Smart Grid is the service provider that distributes power to the consumer. The prosumers formed the Blockchain network, which is a permissioned blockchain network. The blockchain network securely stores the user information and the transactions of the smart meter in the form of blockchain. Our primary contributions are as follows:

1. **Novel Consensus Algorithm:** The proposed novel WNS-BFT consensus algorithm is a fair and robust leader-based consensus algorithm. The WNS-BFT creates the group of honest consensus nodes using two parameters: 1) honest participation score and 2) computational power of the consensus nodes.
2. **Real World Testbed Implementation:** We have developed a testbed using Raspberry Pi devices for conducting experiments. This practical implementation showcases the viability of the proposed protocol in a real-world setting.
3. **Quantitative Performance Metrics:** The *zk-SGB* provides specific performance metrics, such as up to 90% reduction in consensus delay, up to 60% increase in throughput, and up to 80% reduction in computational overhead. These quantified improvements resulting from the use of *zk-SGB* demonstrate the practical benefits of the proposed solution compared with state-of-the-art schemes.
4. **Targeted Security Analysis:** The *zk-SGB* ensures privacy preservation and protection against double-spending, forking, modification attacks, and confidentiality breaches. This targeted security analysis illustrates the challenges specific to energy trading in the smart grid, highlighting the framework's ability to provide robust security in this context.

5.4.5 Consumer and Prosumer Communication Based on zk-SNARK

This section describes the communication between the smart meter acting as the prover and the consensus node acting as the verifier. It is important to note that we have used the zero-knowledge proof cryptographic scheme to ensure the privacy of the user information. The background description of zk-SNARK is described in the Preliminary section. As we have explained in the previous section, the smart meter is connected to the grid operator as well as to the consensus node. It collects various data such as daily power consumption, peak hours of usage, etc. This information is very valuable for a consumer to optimize their power usage and manage their appliance accordingly. Figure 5.24 illustrates the communication process using the zero-knowledge proof, which is defined as follows.

1. Every smart meter is identified by a unique identifier M_{ID}. The smart meter collects all the data called SD_i, i.e., i^{th} collected data. It also records the current timestamp denoted as Tr_1 in the diagram.
2. The hash of all the collected data is calculated as $H(SD_i)$, where $H()$ is the SHA256 hash function. After calculating the hash, it is xor-ed with meter id M_{ID} as depicted in the figure as ($A_1 = H(SD_1) \oplus M_{ID}$). After that, all the transaction is created and signed with the private key of the meter. Finally,

before sending the information to the miner (verifier), the smart meter (prover) creates the proof π where $Tr_1, Tr_2, ...Tr_n$ are treated as information, and A is as a witness and sent to the consensus node with a timestamp as $< \pi, T_1 >$. The transactions are signed with the private key of the meter. Hence these transactions can not be forged in the transmission. Additionally, zero knowledge provides privacy.

3. The consensus node, which receives the transactions from the smart meter, first verifies the proof π and, after that, verifies the signature of signed transactions. It checks whether the received transactions are new or old by checking $\delta T > T_2 - T_1$ where T_2 is the miners' current timestamp. If the condition is true, it further verifies the transaction and creates a block. Here we have considered the transaction-receiving node as the leader node.

4. The leader node verified all the transactions using the public key of the smart meter and forwarded them to other consensus nodes. Then it calculates the Mrt using the function $merkle_tree()$, and PBG and set other parameters as DB_{ID}, Tr_2, and n. After this, the node creates the block, as depicted in the figure, and broadcasts it into the network for verification.

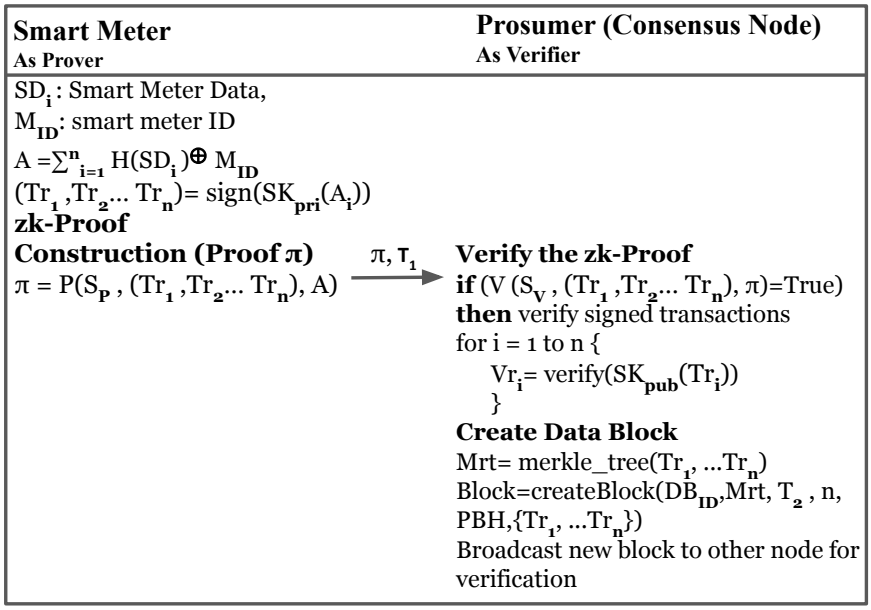


Figure 5.24: Sequence diagram for communication between the consumer (smart meter) and prosumer (consensus node) in *zk-SGB*

5.4.6 WNS-BFT: Consensus Algorithm

The proposed novel Weighted Node Selection Byzantine Fault Tolerance (WNS-BFT) algorithm is the consensus algorithm for permissioned blockchain networks like smart grid energy trading. The WNS-BFT creates the subgroup of consensus nodes out of all the consensus nodes based on the reputation computation. It uses the two important parameters to calculate the Reputation Score (*RS*) of each consensus node, which are the Participation Score (*PS*) and the Computation Power Score (*CPS*). After calculating the *RS* of each consensus node, a subgroup of consensus nodes is created, which is actually the 51% nodes whose *RS* is highest in the whole blockchain network, i.e., amongst all the consensus nodes. This selected subgroup

is responsible for the transaction verification, block generation, block verification, and block commit and append it into the longest chain of the valid blockchain ledger. After appending the newly generated block in the ledger, the *RS* of each node is revised, and the consensus process repeats for the new set of transactions. Let's understand the process in detail.

Let, we have a set of consensus nodes $CN_j = \{CN_1, CN_2, ..., CN_M\}$, where $j = 1, 2, ..., M$. Among these nodes, the CN_x, $x \in j$ is the leader node, which receives the information from the smart meter SM_i. The whole consensus process is divided into three phases, i.e., reputation score computation, fair leader selection, block generation and verification, and node reputation updation of each node.

Step-1: Reputation Score (*RS*) Calculation
To calculate the *RS*, first, we need to calculate the two parameters, i.e., PS_{CN_i} and CPS_{CN_i} of consensus node CN_i, where, $i = 1, 2, ..., M$.

- **Participation Score (PS_{CN_i})**:
  The participation Score (*PS*) represents the involvement of a node in network activities. It can be calculated based on the number of valid blocks the node submits and the frequency of block validations. The PS_{CN_i} of node CN_i expressed as follows:

$$PS_{CN_i} = \frac{Number\ of\ Valid\ Blocks\ from\ CN_i}{Total\ number\ of\ blocks\ in\ the\ chain}$$

- **Computation Power Score (CPS_{CN_i})**:
  The computational Power Score (CPS) represents the computing resources contributed by a node to the network. Similar to the PoW systems, this can be directly related to the node's hashing power. The CPS_{CN_i} of node CN_i expressed as follows:

$$CPS_{CN_i} = \frac{Hashing\ Power\ of\ CN_i}{Total\ hashing\ power\ in\ the\ network}$$

- **Normalization and Scaling**:
  Normalize both participation and computational power scores to be within the range $[0, 1]$.

- **Weighting Factors:**
  Assign weighting factors α and β to *PS* and *CPS*, respectively, to reflect their importance in determining reputation. These weighting factors should be such that $\alpha + \beta = 1$, i.e., these weighting factors control the balance between the two parameters so that only one parameter does not overly inflate the *RS* if the other factor is lower.

- **Weighted Sum Calculation:**
  Calculate the weighted sum (*WS*) of PS_{CN_i} and CPS_{CN_i} as

$$WS = \alpha \cdot PS_{CN_i} + \beta \cdot CPS_{CN_i}$$

- **Limiting Factor (*LF*) Determination:**
  Identify the parameter with the lower value between PS_{CN_i} and CPS_{CN_i} as

$$LF_{CN_i} = min(PS_{CN_i}, CPS_{CN_i})$$

- **Final Reputation Score:**
  Calculate the reputation score (RS_i) using a weighted geometric mean approach (it balances the influence of both parameters) for each node CN_i using the finalized reputation calculation method:

$$RS_{CN_i} = \sqrt[\alpha+\beta]{[WS_{CN_i}]^{\alpha} \cdot [LF_{CN_i}]^{\beta}}$$

Step-2: Fair Leader Selection
In each round of the consensus process, a leader node is selected. First, the consensus nodes (CN) whose reputation score is above or equal to the reputation threshold are selected to form the pre-selected group to make the leader selection process more secure and fair. Second, to make the process fair or prevent the leader node from being known in advance by the attacker, the nodes that satisfy the random number threshold are selected from the pre-selected group. Finally, from these pre-selected consensus nodes, a node with a high reputation score is selected as the leader node. Thus, the leader selection process is a three-step process, i.e., create a pre-selection group, random reputation threshold, and find the leader node.

1. **Create Pre-selection Group**: There are two possible cases.
   (a) **Case 1: If it is the first round of consensus** In this case, the consensus nodes CN_j with $RS_{CN_j} \geq RS$ are selected to form the pre-selection group. Here, the RS_{CN_j} is the node's initial reputation score, and RS is the reputation threshold value.
   (b) **Case 2: Otherwise** In this case, the consensus nodes CN_j with $RS_{CN_j} \geq RS$ are selected to form the pre-selection group. Here, RS_{CN_j} is the updated reputation score of the node CN_j for $j = 1, 2, ..., M$.
   (c) Let, there are $S \in M$ number of consensus nodes in the pre-selection group, i.e., $CN_s = \{CN_1, CN_2, ..., CN_s\}$, $s = 1, 2, ..., S$.
2. **Random Number Threshold**
   Consensus nodes CN_s in the $S \in M$ pre-selection group perform the following computation.
   (a) Select the random number X_s.
   (b) Sign the X_s using ECDSA as $\sigma_{CN_s} = H(X_s)^{Pr_{CN_s}}$, where, Pr_{CN_s} is the private key of CN_s.
   (c) Broadcast this into the other nodes in the pre-selection group.
3. The current leader node CN_x performs the following steps to generate the random number threshold.
   (a) Select the random number X_{CN_x} as the threshold ω.
   (b) Sign the X_{CN_x} using ECDSA as $\sigma_{CN_x} = H(X_{CN_x})^{Pr_{CN_x}}$, where Pr_{CN_x} is the private key of the leader node CN_x.
   (c) Send $< \omega, \sigma_{CN_x} >$ to each node CN_s.
4. Among the consensus nodes group S, the node with the highest reputation score RS_{CN_j} is selected as the leader node CN_x.
5. The information of CN_x is broadcasted to each consensus node group CN_j, $j = 1, 2, ..., M$.

Step-3: Block Generation and Verification:
The block verification scheme is also simple yet very effective. The newly elected leader node sends the newly created block along with the nonce value. Every verifier node again creates the block with the received nonce value and checks the received block's validity. Therefore, the verification process also does not take much time to block verification. These are the two novel things that are introduced in the WNS-BFT. The WNS-BFT uses the following consensus algorithm to create, verify, and add the block to the distributed ledger. We have considered that every consensus node has a transaction pool (TP), where the smart meters of the consumers put all the transactions. The consensus node sends the transaction to other consensus nodes after the transactions are verified. The block generation and validation Algorithm 20 works as follows:

1. The consumer (smart meter) is considered an ordinary node that is responsible for generating the transactions. The consumer is connected to the smart grid, which acts as the consensus node in the *zk-SGB*. All the generated transactions are put into the transaction pool TP. The selected leader node in the blockchain network collects the transactions from the TP and verifies them. After verification, transfer those transactions to other consensus nodes. Then, the leader node calls the function $createHashTree$ to calculate the Merkle root (Mrt) of these transactions. The previous block's hash PBH is calculated using the function $H()$, the current timestamp T, and a nonce (n) are initialized with the appropriate value.

2. Using the values of $Mrt, PBH, T, nonce$, the leader node create the block using $createBlock()$ function. The leader node sends the newly created block in the subgroup (CN_s) for verification. The digital signature is used to prevent the block from modification and impersonation attacks.

3. When the rest of the nodes (other than the leader node) get the copy of the block, first, they verify the signature of the block to check whether it was sent from the legitimate leader node or not. After validating, they also verified the correctness of the block as follows. To validate the block, the verifier nodes recreate the block from the same set of transactions, and for that, it calculates the Mrt, PBG, T, n. After recreating the block, nodes check whether the recreated block matches the received block. If the match is found, the node will set the status of $block_status$ as $Valid$, otherwise set as $Invalid$. In addition to $block_status$, the node will also set the status of leader node $node.status$ as good or bad for the future leader selection process.

4. Suppose the leader node gets at least $2N/3$ responses as $Valid$. In that case, it appends the block in the distributed ledger, i.e., a block is considered valid when half of the nodes verify the block. At least 66% nodes must verify the block to prevent any worst attack.

Step-4: Node Reputation Updation
In *Algorithm* 20, the verifier node responds to the leader node with the status of the block's validity, either 'valid' or 'invalid,' depending on the verification process of the newly proposed block. Alongside the block status, the verifier node also provides the reputation status of the leader node as 'good' or 'bad.' Specifically, if the proposed block from the leader node is deemed valid, the leader node's reputation status is set as 'good'; otherwise, it is set as 'bad.' Accordingly, the participation score (PS) is updated by incrementing the count of valid blocks whenever a newly proposed block is validated. Apart from PS, the computational power score (CPS) is also updated when a consensus node upgrades its computing resources. However, the frequency of updating CPS is significantly lower compared to PS.

Algorithm 20: Block Generation and Verification

Input: Transactions: $Tr_i, i = 1, 2, \ldots, n$
Output: Block
$TP.append(Tr)$;
if *node* = *leader* **then**
    $Mrt =$ createHashTree(TP.getAllTransactions);
    $PBH = H(PB)$;
    $BH = H(Mrt, T, PBH)$;
    $block =$ createBlock(Mrt, T, PBH, BH, n);
    sign($block$);
else
    getBlock($block, nonce$);
    $Mrt =$ createHashTree(TP.getAllTransactions);
    $block^* =$ createBlock(Mrt, T, PBH, BH, n);
    if $block = block^*$ **then**
        setBlockStatus(valid);
        $node$.status = good;
    else
        setBlockStatus(invalid);
        $node$.status = bad;
if $count(block_status = valid) \geq \frac{2N}{3}$ **then**
    $blockchain$.append($block$);

5.4.7 Performance Analysis

The evaluation of *zk-SGB*'s performance revolves around five key parameters: consensus delay, throughput, computational cost, and energy consumption. However, preceding the performance assessment, we provide a comprehensive overview of the implementation environment.

Implementation Environment

In order to bring the proposed scheme into a practical implementation, a carefully designed testbed was constructed. This testbed consisted of a total of 9 Raspberry Pi (RP) boards, with a specific distribution to emulate the roles of various entities within the system. Among these 9 boards, 2 were designated as prosumer nodes, and the remaining 6 RP boards were configured as consumer nodes. Each prosumer was associated with a group of consumers, symbolizing the interaction between residential units in the energy market. The consumer RP boards were designed to replicate the capabilities of smart meters, effectively simulating the data collection and communication functionalities required within the smart grid ecosystem. On the other hand, the prosumer RP boards were implemented as integrated units that combined smart meter capabilities as well as additional features. The choice of Raspberry Pi boards for this implementation stemmed from their computational prowess and their ability to handle the security protocols inherent to the proposed scheme. These RP boards were equipped with microcontrollers capable of processing data and memory units for storing crucial information. By closely mirroring the characteristics of a genuine smart meter, these RP boards became integral components of the testbed. Crucially, these devices were

equipped with Wi-Fi communication capabilities. Two variations of Raspberry Pi boards were employed in this testbed: one with 2GB of memory, which represented the consumer nodes, and the other with 4GB of memory, designated for the prosumer nodes. While the memory capacities differed, both types of boards shared identical software and hardware configurations, which ensured a consistent testing environment. The operating system chosen for all RP boards was Raspberry Pi OS, also known as Raspbian OS, a Debian-based distribution optimized for Raspberry Pi devices. Each RP board was equipped with a robust 1.5GHz Broadcom BCM2711 Quad-core Cortex-A72 processor, enabling efficient data processing and communication. The development of programs for this testbed was carried out using the Java programming language, leveraging its versatility and suitability for implementing the proposed scheme's functionalities.

Consensus Delay and Throughput

During the comprehensive evaluation of *zk-SGB*, various assessment was conducted, encompassing critical performance metrics such as transaction throughput, consensus latency, and the equitable selection of leader nodes. This thorough examination aimed to ascertain the scheme's operational efficiency and resilience while upholding robust security standards. Notably, the evaluation sought to optimize consensus latency and enhance overall throughput without compromising the system's inherent security provisions. The achieved results emphasize the practical viability of these optimizations within the context of the *zk-SGB*.

A particularly noteworthy achievement is the successful reduction of consensus latency and the concurrent elevation of throughput through the novel WNS-BFT consensus algorithm. This optimization was achieved while maintaining the scheme's security measures, reinforcing the notion that performance enhancements need not come at the expense of robust security. The comprehensive analysis of various parameters revealed the superior performance of *zk-SGB* compared with state-of-the-art techniques. It's important to underline that prevailing weaknesses in state-of-the-art techniques are often tied to their susceptibility to the associated high consensus delay and low throughput. These issues can undermine the operational efficiency and reliability of any blockchain-based system. In contrast, the proposed *zk-SGB* stands out as a solution that addresses these issues by implementing an efficient, fair, and robust consensus algorithm WNS-BFT that effectively mitigates the risk of forks and significantly reduces consensus delay. This strategic fortification of the system contributes to its exceptional performance in real-world scenarios. A key architectural decision to reduce propagation delays within the blockchain network involved the deliberate limitation of block size to 1 MB or less. By imposing this restriction, each block can accommodate a maximum of 516 transactions, thereby preventing excessive block sizes that could hinder efficient network propagation. Figure 5.25a clearly demonstrates the notable reduction in consensus delay achieved by *zk-SGB* compared with state-of-the-art schemes. Another significant aspect of the scheme's performance is its throughput, which is aptly showcased in Figure 5.25b. The comparative analysis shows that *zk-SGB* outperforms other blockchain systems in terms of throughput. Unlike traditional systems, where multiple nodes vie for block rewards and simultaneous block creations lead to congestion and forks, *zk-SGB* strategically selects the consensus node as the leader. This approach reduces congestion, increases throughput, and ensures a fair distribution of responsibilities among consensus nodes. In summation, the evaluation of *zk-SGB* highlights its ability to address critical issues in state-of-the-art blockchain methodologies. Through thoughtful optimizations, strategic consensus mechanisms, and innovative leader node selection, *zk-SGB* achieves remarkable gains in consensus latency, throughput, and fairness—all without compromising the high standards of security and reliability demanded by contemporary smart grid applications.

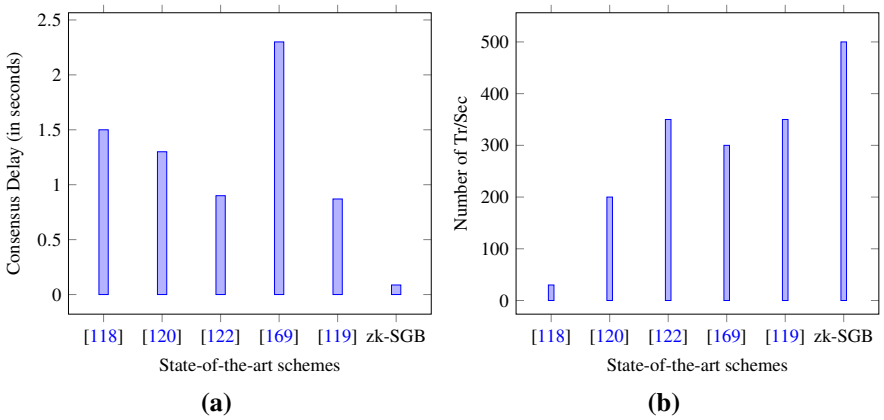


Figure 5.25: Comparative analysis of *zk-SGB* based on (a) consensus delay and (b) throughput with state-of-the-art schemes

Computational Cost and Energy Consumption

The computational cost, denoted as T_{ct} and measured in milliseconds, serves as a fundamental metric to measure the efficiency of the proposed scheme's execution. It encapsulates the time required for various essential operations within the system. These operations encompass critical functions such as the verification of transactions, communication, the creation of new blocks, and the subsequent verification of these blocks. By measuring the computation of these operations, we obtain insights into the overall efficiency of the system's execution. Table 5.13 provides time requirements associated with various cryptographic functions. These functions, including hashing, digital signature generation and verification, encryption, and decryption, are pivotal components of the proposed scheme. Understanding the time allocation for each of these functions offers a detailed perspective on the computational workload involved in the system's operations. Moreover, we evaluate the computational cost of the proposed *zk-SGB* scheme compared with existing schemes. Table 5.18 shows the comparison of the computational cost associated with executing state-of-the-art schemes. This quantitative analysis shows the relative computational efficiency and overhead of the proposed scheme.

Energy consumption is a crucial element in evaluating the practicality and sustainability of a system, especially when it comprises lightweight devices such as smart meters. The energy consumption, denoted as E_{rg}, is derived from the product of computational time (T_{ct}) and the CPU power (C_P), expressed as $E_{rg} = T_{ct} * C_P$. This equation quantifies the total energy consumed by the system during its operation. By considering the CPU power and computational cost, we compute the energy consumption of the system during the whole operation. To visually compare the proposed scheme's computational efficiency against state-of-the-art schemes. This thorough evaluation of computational cost and energy consumption shows the comprehensive perspective on the practical feasibility and efficiency of the proposed *zk-SGB* scheme.

5.4.8 Security Analysis of zk-SGB

This section delves into the security analysis of the *zk-SGB*. The assessment of its security highlights its effectiveness in countering double-spending attacks, preventing forks, ensuring privacy preservation, confidentiality, and prevention from modification. We confirm these assertions through a detailed lemma and

Table 5.18: Comparison of computation cost (CC) in milliseconds and energy consumption (EC) in Joule of *zk-SGB* on Raspberry Pi 4 (RP) 2GB and 4GB with state-of-the-art schemes

Scheme	Computation Cost Equation	CC on RP4 4GB	CC on RP4 2GB	EC on RP4 4GB	EC on RP4 2GB
[122]	$5T_{H(.)} + 3T_{sign} + 3T_{veri}$	17.43	35.97	97.61	86.33
[119]	$6T_{H(.)} + 2T_{sign} + 2T_{veri}$	11.63	24.25	65.128	58.20
[169]	$6T_{H(.)} + 3T_{sign} + 3T_{veri}$	17.44	35.55	97.60	85.32
[170]	$6T_{AES} + 2T_{sign} + 2T_{veri}$	47.36	99.78	265.22	239.48
[118]	$5T_{sign} + 5T_{veri}$	29.00	59.95	162.4	143.88
[120]	$11T_{H(.)} + 8T_{mul(.)}$	40.05	55.49	224.30	133.18
zk-SGB	$3T_{H(.)} + 1T_{sign} + 1T_{veri}$	5.82	11.89	32.56	28.54

proof that clarifies the mechanisms that safeguard the system from the threats mentioned above. The smart grid's security capabilities have been strengthened against cyberattacks through a blockchain-based security framework, as proposed by Liang et al. [118]. Their approach employed a PoW-based consensus mechanism devoid of any reward system. While mining was a competitive process, there was no reward incentive for the miner who solved the puzzle first. Although the scheme appeared resilient against double-spending attacks, it shows vulnerabilities related to forking and privacy attacks. Similarly, the Guardian scheme proposed in [119] also adopted the PoW consensus algorithm. Although it was adept at protecting against double-spending attacks, it fell short in protecting against forking and privacy attacks. Additionally, fault tolerance was not a characteristic of this scheme. A smart contract-based smart grid system was put forth in [120]. This scheme facilitated anonymous authentication; however, it remained susceptible to double-spending and lacked robust fault tolerance. Addressing the issues of double spending and forking, the scheme proposed in [122]. However, when it came to a substantial number of nodes (prosumers and consumers), it revealed its inability to maintain fault tolerance [171]. In contrast, the *zk-SGB* proposed a novel consensus algorithm called WNS-BFT. This effectively thwarts all four attacks and performs better than the state-of-the-art techniques. With the help of 3 lemmas below, we have described how the *zk-SGB* is secured against the double-spending attack and fork attack and ensured the preservation.

Lemma 22. *Let Tr_1 and Tr_2 be two conflicting transactions aiming to spend the same energy token within the zk-SGB framework. The objective is that the probability of both Tr_1 and Tr_2 being successfully validated and added to the blockchain is negligible. The zk-SGB, incorporating the WNS-BFT consensus algorithm, prevents such double-spending attacks.*

Proof. Assumption: Both transactions, Tr_1 and Tr_2, are simultaneously submitted to the *zk-SGB* network in an attempt to execute a double-spending attack. Each transaction T_i $(i = 1, 2)$ undergoes a consensus process involving N consensus nodes. To be deemed valid, a transaction requires approval from at least $2N/3$ nodes. Let p_v be the probability that a single consensus node validates transaction Tr_i. Given the requirement of $2N/3$ approvals for validity, the probability of Tr_i being successfully validated is

$$p_{\text{valid}} = \binom{N}{2N/3} \cdot p_v^{2N/3} \cdot (1 - p_v)^{1N/3}.$$

Conflicting transactions, Tr_1 and Tr_2 compete to be added to the blockchain, but only one can prevail due to their conflicting nature. The probability of a double-spending attack succeeding is the probability that both Tr_1 and Tr_2 are successfully validated and added to the blockchain. This probability can be bounded by the probability of individual validation p_{valid} and the assumption that only one transaction will prevail in the blockchain.

Let p_m be the probability that a consensus node behaves maliciously and approves both Tr_1 and Tr_2. Considering the competitive and reputation-based consensus node selection process, it is reasonable to assume that the majority of nodes will not engage in such malicious behavior. Thus, p_m is expected to be negligible. The probability of the double-spending attack succeeding is bounded by $p_{\text{attack}} = p_{\text{valid}} \cdot p_m$, which is negligible due to the negligible nature of p_m. Thus, by combining the negligible probabilities of both successful validation of conflicting transactions and malicious behavior, it can be concluded that the probability of a double-spending attack succeeding in the *zk-SGB* is negligible. However, it's important to emphasize that while this proof offers a probabilistic perspective on the prevention of double-spending attacks, the practical implications may vary due to factors beyond the scope of this simplified analysis. The actual behavior of consensus nodes, network conditions, and other considerations can influence the true probabilities in real-world scenarios. □

Lemma 23. *Let P_{fork} be the fork probability in the WNS-BFT consensus algorithm. The P_{fork} is almost negligible in the zk-SGB under the consensus algorithm WNS-BFT.*

Proof. An attacker attempts to create a fork by manipulating the leader selection process and controlling consensus nodes. The fork probability P_{fork} can be defined as:

$$P_{\text{fork}} = \frac{\textit{Number of Successful Forks}}{\textit{Total Number of Consensus Rounds}}$$

Each consensus node CN_i is assigned a reputation score R_i using a weighted geometric mean approach:

$$R_i = \sqrt[\alpha+\beta]{[WS_{CN_i}]^{\alpha} \cdot [LF_{CN_i}]^{\beta}}$$

where $WS_{CN_i} = \alpha \cdot PS_{CN_i} + \beta \cdot CPS_{CN_i}$ and $\alpha + \beta = 1$. The attacker must maintain a consistently high reputation R_i based on both PS_{CN_i} and CPS_{CN_i}. Now Let X_s be the random number chosen by consensus node CN_s and σ_{CN_s} its corresponding signature. The attacker would need to predict a threshold ω generated by the leader CN_x and broadcasted as $< \omega, \sigma_{CN_x} >$. This cryptographic unpredictability makes prediction unfeasible. When a leader CN_x is selected, its reputation status is broadcasted. The attacker's reputation status as "bad" would be known if they deviate from consensus rules, undermining their credibility. Verifier nodes collectively validate the proposed block. The attacker's malicious block proposal would be rejected by the majority, preventing a fork. Additionally, the *CPS* update frequency is low compared to *PS*, preventing rapid manipulation of CPS_{CN_i} by an attacker.

Considering reputation-based selection, random threshold, reputation broadcast, collective verification, and limited *CPS* update frequency, we conclude that the fork probability P_{fork} is almost negligible. The combination of reputation, cryptographic unpredictability, transparency, collective verification, and limited influence of *CPS* fortifies the consensus mechanism against fork attempts. □

Lemma 24. *Let $\mathbb{M}$ denote the set of smart meters and $\mathbb{V}$ denote the set of verifier nodes (consensus nodes). zk-SGB maintains privacy and security through zk-SNARK, a combination of reputation-based leader selection, cryptographic techniques, and communication protocols. Thus, zk-SGB ensures privacy preservation and protects against modification and confidentiality attacks.*

Proof. The *zk-SGB* scheme incorporates zero-knowledge proof (zk-SNARK) cryptographic techniques in the communication between smart meters and verifier nodes. These techniques ensure that sensitive user information remains private during the communication process. The steps are as follows:

For each smart meter $M_i \in \mathbb{M}$, SD_i represents the collected data, and Tr_1 is the timestamp of communication. The smart meter generates a proof π_i using zk-SNARK after converting the smart meter data into transactions Tr_i, $i = 1,2,3,...,n$: $\pi_i = P(S_P, Tr_i, T_1)$. For each verifier node $V_j \in \mathbb{V}$, the verifier receives the proof π_i, transactions, and timestamp Tr_1 from the smart meter. The verifier validates the proof using zk-SNARK verification: $verify(S_v, Tr_i, T_1, ..., \pi_i)$. The zk-SNARK cryptographic scheme guarantees that the verifier only learns the validity of transactions without gaining access to actual transaction details. This ensures user privacy while allowing the network to verify the integrity of transactions. Additionally, the *zk-SGB* is also protection against modification and confidentiality attacks. This can be expressed as follows. Transactions created by smart meters are signed using their private keys Pr_{M_i}. Let Tr_i denote the transaction signed by smart meter M_i. This prevents unauthorized modification of transactions during transmission: $Tr_i = \text{Sign}(SD_i, Pr_{M_i})$. For each verifier node $V_j \in \mathbb{V}$, the verifier checks the timestamp of received transactions to prevent replay attacks:$T_1 \leq T_{\text{current}}$. Reputation-based leader selection ensures that only nodes with high reputation scores become leaders. Let R_{M_i} denote the reputation score of smart meter M_i. The leader selection process involves selecting nodes from $\mathbb{V}$ that satisfy reputation and random number thresholds. This prevents malicious nodes from taking control of the consensus process. The combination of privacy-preserving zk-SNARK techniques, data integrity through digital signatures, timestamp verification, reputation-based leader selection, and collective verification ensures that the *zk-SGB* maintains privacy, guards against modification and confidentiality attacks and provides a robust consensus mechanism for the smart grid. □

Concluding Remark

In this Chapter, we explored the application of blockchain technology within the context of smart cities, focusing on three major domains: IoT, Intelligent Transportation Systems (ITS), and Smart Grids. For IoT, we introduced BLISS and SCAB-IOTA, which enhance data integrity and scalability in IoT networks. In the domain of ITS, LBSV and SmartCoin were developed to secure communication and enable efficient micropayments within intelligent transportation networks. For the Smart Grid, SmartGrid-NG, ETradeChain, and zk-SGB were proposed to facilitate secure energy trading and efficient grid management. Each technique was designed to address specific challenges within its respective domain, demonstrating the versatility and potential of blockchain in improving various aspects of smart city infrastructure. This chapter has illustrated the practical benefits of applying blockchain technology to real-world problems, particularly in enhancing security, efficiency, transparency, and privacy. In the final chapter of this thesis, we will summarize the key contributions made across all chapters and discuss their broader implications for the future of blockchain technology, along with the future directions.

6

Conclusion and Future Work

CONCLUSION

This thesis provides a comprehensive investigation into the expansive capabilities of blockchain technology, extending its applications well beyond the confines of cryptocurrency. Chapter 1 introduces the foundational concepts of blockchain, setting the stage for the in-depth exploration that follows. Chapter 2 thoroughly reviews existing literature, highlighting key challenges and gaps in current blockchain research. This contextual foundation leads to the subsequent contributions, where novel solutions are proposed to address critical issues such as scalability, interoperability, and domain-specific applications of blockchain. Chapter 3 focuses on the critical issue of scalability in public blockchains by introducing two novel sharding-based blockchain protocols, namely BlockTree and UnoShard. BlockTree implements a hierarchical structure that divides the network based on the number of miners in each group, significantly enhancing transaction handling and scalability. This structure allows for efficient block validation and propagation, mitigating common bottlenecks that typically hinder scalability. On the other hand, UnoShard leverages DPoS to divide the network into shards, distributing the workload across these shards to enable effective scaling without compromising security. It also ensures fair shard allocation, load distribution, and efficient intra-shard and cross-shard consensus to provide a robust solution to the scalability challenges. Our comparative analysis reveals that both BlockTree and UnoShard outperform state-of-the-art scalability solutions, offering promising directions for the future of scalable public blockchains. Chapter 4 then transitions the focus from scalability to interoperability, introducing the CrossLedger protocol. CrossLedger is designed to enable secure and efficient cross-chain asset transfers, addressing the complexities of interoperability in multi-blockchain environments. By ensuring essential features such as non-repudiation, unlinkability, confidentiality, atomicity, and interoperability, CrossLedger paves the way for a more interconnected and versatile blockchain ecosystem.

Chapter 5 explores blockchain applications within the context of smart cities, focusing on IoT, ITS, and Smart Grid. Each section of this chapter presents independent solutions tailored to the specific challenges of these domains. For IoT, BLISS and SCAB-IoTA enhance communication security and data integrity. In ITS, LBSV and SmartCoin provide blockchain-based solutions for secure vehicular communication and incentivized data sharing. The mPBFT consensus algorithm in LBSV and LCA in SmartCoin both demonstrate superior performance compared to existing state-of-the-art solutions. In the Smart Grid domain, SmartGrid-NG, ETradeChain, and zk-SGB offer innovative approaches to energy management and decentralized energy trading. Each of these systems introduces novel consensus algorithms, such as GridChain

in SmartGrid-NG, a two-level consensus mechanism in ETradeChain, and the WNS-BFT in zk-SGB, all of which enhance the fairness, efficiency, and security of their respective applications.

Overall, this thesis emphasizes the vast potential of blockchain technology across various domains. By addressing critical challenges in scalability, interoperability, and application-specific implementations, this work contributes to the advancement of blockchain technology and its integration into the digital infrastructure. The findings of this thesis highlight the transformative impact blockchain can have on diverse fields, providing a strong foundation for future research and innovation in this rapidly evolving area.

FUTURE DIRECTION

Future research in blockchain technology could significantly benefit from focusing on enhancing interoperability across diverse blockchain platforms, building on the foundation established by CrossLedger. While CrossLedger has enabled cross-chain communication, there is still considerable potential for creating more adaptable and universal protocols that facilitate seamless interactions across a broader range of blockchain networks. Additionally, the scalability improvements achieved through BlockTree and UnoShard demonstrate the promise of sharding techniques. In these approaches, the focus has been on the efficient division of the network to enhance throughput and security; however, future efforts should also explore the efficient placement of transactions within different shards to further improve throughput and reduce latency. Moreover, integrating blockchain with emerging technologies such as redactable blockchain and quantum computing presents an exciting frontier. By combining these technologies, future research could unlock new possibilities for creating even more powerful, scalable, and versatile blockchain applications that address a wide range of real-world challenges.

Redactable Blockchain is one of the growing trends in the field of blockchain optimization. The traditional blockchains are immutable, which is its core principle. While this principle is a crucial pillar of blockchain security, it also presents challenges in situations where data must be altered or deleted, such as in the case of errors, obsolete information, or compliance with data protection regulations like the General Data Protection Regulation (GDPR) in Europe or IT ACT in Bharat. The redactable blockchain concept has the potential to address all these issues by allowing specific data in the blockchain ledger to be altered or deleted under strict conditions without compromising the overall integrity and security of the blockchain. Designing an efficient redactable blockchain scheme could open new avenues for blockchain applications such as IoT, ITS, and Smart Grid.

The rapid growth of quantum computing poses a serious challenge to traditional blockchain technology. Parallelly it also provides opportunities. It poses challenges because quantum computers, with their ability to solve complex mathematical problems, could potentially break the traditional cryptographic algorithms, such as SHA and ECC, that underpin the security of blockchain systems. Due to this vulnerability of cryptographic algorithms used in blockchain, the demand for quantum-resistant cryptographic techniques is much higher so that they can protect the blockchain against quantum attacks. Additionally, quantum computing could also be leveraged to create new types of blockchain systems—quantum blockchains—that exploit quantum entanglement and other quantum phenomena to achieve unprecedented levels of security, efficiency, and scalability. In future research, we will explore all the possibilities of quantum blockchain deployment in the real world scenarios.

Publications and Achievements

This appendix lists important publications that resulted from the methods and solutions I developed during my PhD research. These papers have undergone peer review and have been shared at reputed transactions, journals and conferences, confirming the new techniques and applications I created in my research on blockchain scalability, interoperability, and different applications. We have included the findings from these publications, particularly in the sections discussing methods and results. Along with the publications, we have also included the list of honours and achievements that we have made during the whole journey of the PhD program.

TRANSACTIONS

1. **Vishwakarma, Lokendra**, Ankur Nahar, and Debasis Das. "LBSV: Lightweight Blockchain Security Protocol for Secure Storage and Communication in SDN-Enabled IoV." **IEEE Transactions on Vehicular Technology** 71, no. 6 (2022): 5983-5994, **Impact Factor: 6.8**

2. Kumar, Amritesh, **Lokendra Vishwakarma**, and Debasis Das. "LandChain: A MultiChain Based Novel Secure Land Record Transfer System." **IEEE Transactions on Reliability** (2024), **Impact Factor: 5.9**

3. Barbhaya, Umang Rajendra, **Lokendra Vishwakarma**, and Debasis Das. "ETradeChain: Blockchain-Based Energy Trading in Local Energy Market (LEM) Using Modified Double Auction Protocol." **IEEE Transactions on Green Communications and Networking** (2023), **Impact Factor: 4.8**

JOURNALS

1. **Vishwakarma, Lokendra**, and Debasis Das. "SCAB-IoTA: Secure communication and authentication for IoT applications using blockchain." **Journal of Parallel and Distributed Computing** 154 (2021): 94-105, **Impact Factor: 3.8**

2. **Vishwakarma, Lokendra**, and Debasis Das. "SmartCoin: A novel incentive mechanism for vehicles in intelligent transportation system based on consortium blockchain." **Vehicular Communications** 33 (2022): 100429, **Impact Factor: 6.7**

3. **Vishwakarma, Lokendra**, and Debasis Das. "BlockTree: a nonlinear structured, scalable and distributed ledger scheme for processing digital transactions." **Cluster Computing** 24, no. 4 (2021): 3751-3765, **Impact Factor: 4.4**

4. **Vishwakarma, Lokendra**, and Debasis Das. "BLISS: Blockchain-based Integrated Security System for Internet of Things (IoT) Applications." **International Journal of Information Security** (2024): 1-17, **Impact Factor: 3.2**

5. Gupta Shristy, Amritesh Kumar, **Lokendra Vishwakarma**, Debasis Das, "Enhancing Blockchain Scalability and Security: The Early Fraud Detection (EFD) Framework for Optimistic Rollups", **Cluster Computing**, 2024, **Impact Factor: 4.4**

6. Kumar, Amritesh, **Lokendra Vishwakarma**, and Debasis Das. "R-PBFT: A secure and intelligent consensus algorithm for Internet of vehicles." **Vehicular Communications** 41 (2023): 100609, **Impact Factor: 6.7**

CONFERENCES PROCEEDINGS

1. **Vishwakarma, Lokendra**, Amritesh Kumar, Jeevan Madugunda, Debasis Das. "FinBlock: Secure and Fast Transaction Confirmation with High Throughput for FinTech Application" In **IEEE Wireless Communications and Networking Conference 21–24 April 2024, Core Ranking: B**

2. **Vishwakarma, Lokendra**, Debasis Das, Sajal K. Das, and Christian Becker. "SmartGrid-NG: Blockchain Protocol for Secure Transaction Processing in Next Generation Smart Grid." In Proceedings of the 25th **International Conference on Distributed Computing and Networking**, pp. 174-185. 2024, **Core Ranking: B**.

3. **Vishwakarma, Lokendra**, Amritesh Kumar, and Debasis Das. "CrossLedger: A Pioneer Cross-chain Asset Transfer Protocol." In 2023 IEEE/ACM 23rd **International Symposium on Cluster, Cloud and Internet Computing (CCGrid)**, pp. 568-578. IEEE, 2023, **Core Ranking: A**

4. Kumar, Amritesh, Monu Nagar, **Lokendra Vishwakarma**, and Debasis Das. "HN-mPBFT: A Healthy Node based Modified Secure and Fast Consensus Mechanisms for Internet of Vehicles." In 2023 **International Wireless Communications and Mobile Computing (IWCMC)**, pp. 1638-1643. IEEE, 2023, **Core Ranking: B**

5. Nahar, Ankur, **Lokendra Vishwakarma**, and Debasis Das. "Metoids: Hybrid k-medoids-meta heuristic clustering-based routing optimization in vehicular ad-hoc networks." In 2022 IEEE 95th **Vehicular Technology Conference:(VTC2022-Spring)**, pp. 1-5. IEEE, 2022, **Core Ranking: B**

6. **Vishwakarma, Lokendra**, and Debasis Das. "BSS: Blockchain Enabled Security System for Internet of Things Applications." In 2020 IEEE 19th **International Symposium on Network Computing and Applications (NCA)**, pp. 1-4. IEEE, 2020, **Core Ranking: B**

COMMUNICATED

1. **Lokendra Vishwakarma**, Amritesh Kumar, Debasis Das. "UnoShard: Harmonizing Scalability and Security in Blockchain Sharding." In **IEEE Transactions on Information Forensics and Security (TIFS), 2024, Impact Factor: 6.8**

2. **Lokendra Vishwakarma**, Debasis Das. "zk-SGB: Privacy-Preserved Blockchain Framework for Energy Trading in Smart Grid Using Zero Knowledge Proof." In **IEEE Transactions on Consumer Electronics, Impact Factor: 4.3**.

3. **Lokendra Vishwakarma**, Umang Rajendra Barbhaya, Debasis Das, Sukumar Nandi. "GreenChain: Utilizing Consortium Blockchain for Peer-to-Peer Energy Trading in Smart Grid." In **ACM 31st**

Annual International Conference on Mobile Computing and Networking (MobiCom) Nov 2025, Hong Kong, China, Core Ranking: A*

4. Amritesh Kumar, **Lokendra Vishwakarma**, Debasis Das. "GB^2S: A Generalized Balanced Blockchain System." **IEEE Transactions on Reliability 2024, Impact Factor: 5**

5. Amritesh Kumar, Shristy Gupta, **Lokendra Vishwakarma**, Debasis Das. "CollyBus: A Novel Framework for Cross Border Transactions in Blockchain." In **22nd International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks. WiOpt 2024**

HONORS AND ACHIEVEMENTS

1. Received **SERB International Travel Support Grant** for participating in the IEEE Wireless Communications and Networking Conference (WCNC) 2024 at Dubai, United Arab Emirates, by the Science and Engineering Research Board, DST India.

2. Received **ACM India Research Facilitation Grant (RFG) 2024** for the overlength page charges for the publication in the **IEEE Transactions on Green Communications and Networking.**

3. Article Selected in the **ACM Academic Research and Careers for Students (ARCS) 2024** for the paper and poster presentation.

4. **Student Travel Award (STA)** for participating in the **ACM ICDCN 2024** in Jan 2024 and **ACM/IEEE CCGRID 2023** in May 2023 by CCGRID Student Grant Committee.

5. Published research article "Blockchain-based Security Protocol for Next-Generation Smart Grid Transactions." in **TechScape, 2020, Indian Institute of Technology Jodhpur**.

6. Awarded best **Teaching Assistantship (TA)** in the years 2022 and 2023.

7. Held the position of Chair at ACM Student Chapter IIT Jodhpur from July 2021 to July 2022.

8. Published Poster "pzLedger: Privacy Preservation for Energy Trading Transaction in Smart Grid" in **International Conference on Information Systems Security (ICISS 2022), National Conference**, IIT Tirupati, India.

9. Received **Student Travel Grant** from Science and Engineering Research Board (SERB), **Department of Science and Technology (DST), India**, for participating in Karyashala on Vehicular Communications for Next-Generation Intelligent Transportation Systems, held at Indian Institute of Technology Indore, July 2022.

10. Published Poster "SmartCoin: A novel incentive mechanism for vehicles in intelligent transportation system based on consortium blockchain" in **Network and Distributed System Security (NDSS) 2022 (Core Ranking: A*)**, San Diego, California, USA.

11. Received **Student Registration Grant** for participating in the **USENIX Security** August 2021 and **ACM SIGMETRICS**, June 2021.

12. Won the **first prize** in the Student's Pitch Competition in the category of Smart Infrastructure held as part of **Industry Day 2021**, organized at the **Indian Institute of Technology Jodhpur**, Rajasthan.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://assets.pubpub.org/d8wct41f/31611263538139.pdf.

[2] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, "A survey of state-of-the-art on blockchains: Theories, modelings, and tools," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–42, 2021.

[3] B.-J. Butijn, D. A. Tamburri, and W.-J. v. d. Heuvel, "Blockchains: A systematic multivocal literature review," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–37, 2020.

[4] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[5] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[6] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[7] J. Sengupta, S. Ruj, and S. D. Bit, "A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot," *Journal of Network and Computer Applications*, vol. 149, p. 102 481, 2020.

[8] B. Cao, Z. Wang, L. Zhang, D. Feng, M. Peng, L. Zhang, and Z. Han, "Blockchain systems, technologies, and applications: A methodology perspective," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 353–385, 2023. DOI: 10.1109/COMST.2022.3204702.

[9] J. Kolb, M. AbdelBaky, R. H. Katz, and D. E. Culler, "Core concepts, challenges, and future directions in blockchain: A centralized tutorial," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–39, 2020.

[10] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*, Springer, 1987, pp. 369–378.

[11] *Delegated proof of stake (dpos)*, https://how.bitshares.works/en/master/technology/dpos.html, 2019.

[12] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[13] S. Islam, M. J. Islam, M. Hossain, S. Noor, K.-S. Kwak, and S. M. R. Islam, "A survey on consensus algorithms in blockchain-based applications: Architecture, taxonomy, and operational issues," *IEEE Access*, vol. 11, pp. 39 066–39 082, 2023. DOI: 10.1109/ACCESS.2023.3267047.

[14] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, no. 1, 2012.

[15] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[16] C. Sendner, H. Chen, H. Fereidooni, L. Petzi, J. König, J. Stang, A. Dmitrienko, A.-R. Sadeghi, and F. Koushanfar, "Smarter contracts: Detecting vulnerabilities in smart contracts with deep transfer learning.," in *NDSS*, 2023.

[17] C. Jentzsch, "Decentralized autonomous organization to automate governance," *White paper, November*, 2016.

[18] *A blockchain platform for the enterprise - hyperledger fabric*, https://hyperledger-fabric.readthedocs.io/en/release-2.5/, 2023.

[19] *The corda platform: An introduction*, https://corda.net/content/corda-platform-whitepaper.pdf, 2018.

[20] *Quorum - enterprise ethereum client*, https://goquorum.readthedocs.io/, 2019.

[21] M. Li, Y. Lin, J. Zhang, and W. Wang, "Cochain: High concurrency blockchain sharding via consensus on consensus," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10. DOI: 10.1109/INFOCOM53939.2023.10228892.

[22] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[23] R. Han, J. Yu, H. Lin, S. Chen, and P. Esteves-Veríssimo, "On the security and performance of blockchain sharding," *Cryptology ePrint Archive*, 2021.

[24] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93 039–93 054, 2022. DOI: 10.1109/ACCESS.2022.3200051.

[25] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[26] Y. Liu, X. Xing, H. Cheng, D. Li, Z. Guan, J. Liu, and Q. Wu, "A flexible sharding blockchain protocol based on cross-shard byzantine fault tolerance," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2276–2291, 2023. DOI: 10.1109/TIFS.2023.3266628.

[27] A. Lohachab, S. Garg, B. Kang, M. B. Amin, J. Lee, S. Chen, and X. Xu, "Towards interconnected blockchains: A comprehensive review of the role of interoperability among disparate blockchains," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–39, 2021.

[28] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–41, 2021.

[29] J. Ping, Z. Yan, and S. Chen, "A privacy-preserving blockchain-based method to optimize energy trading," *IEEE Transactions on Smart Grid*, vol. 14, no. 2, pp. 1148–1157, 2023. DOI: 10.1109/TSG.2022.3198165.

[30] X. Hu, J. Yin, G. Cheng, J. Gong, L. Yang, H. Chen, and Z. Liang, "Verifying privacy-preserving financing orders on a consortium blockchain based on zk-snarks," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 1527–1532. DOI: 10.1109/WCNC51071.2022.9771773.

[31] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*, IEEE, 2014, pp. 459–474.

[32] V. Buterin, *Zk-snarks: Under the hood*, https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6, 2018.

[33] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*, 2019, pp. 203–225.

[34] D. Mechkaroska, V. Dimitrova, and A. Popovska-Mitrovikj, "Analysis of the possibilities for improvement of blockchain technology," in *2018 26th Telecommunications Forum (TELFOR)*, 2018, pp. 1–4. DOI: 10.1109/TELFOR.2018.8612034.

[35] M. Basile, G. Nardini, P. Perazzo, and G. Dini, "Segwit extension and improvement of the blocksim bitcoin simulator," in *2022 IEEE International Conference on Blockchain (Blockchain)*, 2022, pp. 115–123. DOI: 10.1109/Blockchain55522.2022.00025.

[36] T. Neudecker and H. Hartenstein, "Short paper: An empirical analysis of blockchain forks in bitcoin," in *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, Springer, 2019, pp. 84–92.

[37] Y. Kwon, H. Kim, J. Shin, and Y. Kim, "Bitcoin vs. bitcoin cash: Coexistence or downfall of bitcoin cash?" In *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 935–951. DOI: 10.1109/SP.2019.00075.

[38] A. N. Islam, M. Mäntymäki, and M. Turunen, "Why do blockchains split? an actor-network perspective on bitcoin splits," *Technological Forecasting and Social Change*, vol. 148, p. 119 743, 2019.

[39] X. Yi, Y. Fang, D. Wu, and L. Jiang, "Blockscope: Detecting and investigating propagated vulnerabilities in forked blockchain projects," in *NDSS*, 2023.

[40] D. Li, W. Peng, W. Deng, and F. Gai, "A blockchain-based authentication and security mechanism for iot," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2018, pp. 1–6.

[41] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future generation computer systems*, vol. 88, pp. 173–190, 2018.

[42] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in iot: The challenges, and a way forward," *Journal of Network and Computer Applications*, vol. 125, pp. 251–279, 2019.

[43] Z. Cui, X. Fei, S. Zhang, X. Cai, Y. Cao, W. Zhang, and J. Chen, "A hybrid blockchain-based identity authentication scheme for multi-wsn," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 241–251, 2020.

[44] *Vital stats: Overview of road accidents in india*, https://www.prsindia.org/policy/vital-stats/overview-road-accidents-india, Accessed: 2017.

[45] *Road accidents in india 2020 report*, https://morth.nic.in/road-accident-in-india, Accessed: 2020.

[46] *Who report: Road traffic injuries*, https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, Accessed: 2018.

[47] S. Sharma and B. Kaushik, "A survey on internet of vehicles: Applications, security issues & solutions," *Vehicular Communications*, vol. 20, p. 100 182, 2019.

[48] X. Xu, D. Zhu, X. Yang, S. Wang, L. Qi, and W. Dou, "Concurrent practical byzantine fault tolerance for integration of blockchain and supply chain," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–17, 2021.

[49] J. Bao, D. He, M. Luo, and K.-K. R. Choo, "A survey of blockchain applications in the energy sector," *IEEE Systems Journal*, 2020.

[50] M. F. Munoz, K. Zhang, and F. Amara, "Zipzap: A blockchain solution for local energy trading," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2022, pp. 1–5. DOI: 10.1109/ICBC54727.2022.9805486.

[51] R. Myrzashova, S. H. Alsamhi, A. V. Shvetsov, A. Hawbani, and X. Wei, "Blockchain meets federated learning in healthcare: A systematic review with challenges and opportunities," *IEEE Internet of Things Journal*, pp. 1–1, 2023. DOI: 10.1109/JIOT.2023.3263598.

[52] S. H. Alsamhi, A. V. Shvetsov, S. V. Shvetsova, A. Hawbani, M. Guizani, M. A. Alhartomi, and O. Ma, "Blockchain-empowered security and energy efficiency of drone swarm consensus for environment exploration," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 328–338, 2023. DOI: 10.1109/TGCN.2022.3195479.

[53] Z. Hussein, M. A. Salama, and S. A. El-Rahman, "Evolution of blockchain consensus algorithms: A review on the latest milestones of blockchain consensus algorithms," *Cybersecurity*, vol. 6, no. 1, p. 30, 2023.

[54] P. Vasin, "Blackcoin's proof-of-stake protocol v2," *URL: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf*, vol. 71, 2014.

[55] M. Fitzi, P. Gazi, A. Kiayias, and A. Russell, "Proof-of-stake blockchain protocols with near-optimal throughput.," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 37, 2020.

[56] A. Kiayias, S. Quader, and A. Russell, "Consistency of proof-of-stake blockchains with concurrent honest slot leaders," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2020, pp. 776–786.

[57] V. Bagaria, A. Dembo, S. Kannan, S. Oh, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Proof-of-stake longest chain protocols: Security vs predictability," in *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, 2022, pp. 29–42.

[58] D. Karakostas and A. Kiayias, "Securing proof-of-work ledgers via checkpointing," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2021, pp. 1–5.

[59] M. Kohlweiss, V. Madathil, K. Nayak, and A. Scafuro, "On the anonymity guarantees of anonymous proof-of-stake protocols," in *2021 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2021, pp. 1818–1833.

[60] A. Song, Y. Dai, R. Ji, and Z. Song, "Extend pbft protocol with l-dag," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 2021, pp. 963–970. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00135.

[61] K. Antoniadis, J. Benhaim, A. Desjardins, E. Poroma, V. Gramoli, R. Guerraoui, G. Voron, and I. Zablotchi, "Leaderless consensus," *Journal of Parallel and Distributed Computing*, vol. 176, pp. 95–113, 2023.

[62] S. Liu, R. Zhang, C. Liu, C. Xu, and J. Wang, "An improved pbft consensus algorithm based on grouping and credit grading," *Scientific Reports*, vol. 13, no. 1, p. 13 030, 2023.

[63] Y. Xu and Y. Huang, "An n/2 byzantine node tolerate blockchain sharding approach," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 349–352.

[64] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th {usenix} security symposium ({usenix} security 16)*, 2016, pp. 279–296.

[65] J. Liu, L. Yan, and D. Wang, "A hybrid blockchain model for trusted data of supply chain finance," *Wireless personal communications*, pp. 1–25, 2021.

[66] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[67] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[68] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 375–392.

[69] D. D. F. Maesa and P. Mori, "Blockchain 3.0 applications survey," *Journal of Parallel and Distributed Computing*, 2020.

[70] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*, 2016, pp. 45–59.

[71] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[72] M. Tran, L. Luu, M. S. Kang, I. Bentov, and P. Saxena, "Obscuro: A bitcoin mixer using trusted execution environments," in *Proceedings of the 34th Annual Computer Security Applications Conference*, New York, NY, USA: Association for Computing Machinery, 2018, pp. 692–701.

[73] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 {USENIX} Annual Technical Conference ({USENIX} {ATC} 20)*, 2020, pp. 515–528.

[74] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018, pp. 583–598.

[75] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 95–112.

[76] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 31–42.

[77] T. Crain, C. Natoli, and V. Gramoli, "Red belly: A secure, fair and scalable open blockchain," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 466–483.

[78] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18, Toronto, Canada: Association for Computing Machinery, 2018, pp. 931–948, ISBN: 9781450356930.

[79] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 17–30.

[80] A. Mustafa, S. Alberto, B. Shehar, H. Dave, and D. George, "Chainspace: A sharded smart contracts platform," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, USENIX, 2018. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018%5C_09-2%5C_Al-Bassam%5C_paper.pdf.

[81] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer pbft consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2021. DOI: 10.1109/TPDS.2020.3042392.

[82] G. A. Marson, S. Andreina, L. Alluminio, K. Munichev, and G. Karame, "Mitosis: Practically scaling permissioned blockchains," in *Proceedings of the 37th Annual Computer Security Applications Conference*, 2021, pp. 773–783.

[83] P. Gazi, L. Ren, and A. Russell, "Practical settlement bounds for proof-of-work blockchains," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1217–1230.

[84] Z. Cai, J. Liang, W. Chen, Z. Hong, H.-N. Dai, J. Zhang, and Z. Zheng, "Benzene: Scaling blockchain with cooperation-based sharding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 639–654, 2022.

[85] B. Yu, H. Zhao, T. Zhou, N. Sheng, X. Li, and J. Xu, "Overshard: Scaling blockchain by full sharding with overlapping network and virtual accounts," *Journal of Network and Computer Applications*, vol. 220, p. 103 748, 2023.

[86] P. K. Singh, R. Singh, S. K. Nandi, K. Z. Ghafoor, D. B. Rawat, and S. Nandi, "Blockchain-based adaptive trust management in internet of vehicles using smart contract," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3616–3630, 2020.

[87] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19, London, United Kingdom: Association for Computing Machinery, 2019, pp. 585–602, ISBN: 9781450367479.

[88] G. Wang, S. Wang, V. Bagaria, D. Tse, and P. Viswanath, "Prism removes consensus bottleneck for smart contracts," in *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*, IEEE, 2020, pp. 68–77.

[89] L. Yang, V. Bagaria, G. Wang, M. Alizadeh, D. Tse, G. Fanti, and P. Viswanath, "Prism: Scaling bitcoin by 10,000 x," *arXiv preprint arXiv:1909.11261*, 2019.

[90] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: Communication across distributed ledgers.," *Cryptology ePrint Archive*, vol. 12675, 2019.

[91] J. Kwon and E. Buchman, "A network of distributed ledgers," *Cosmos, dated*, pp. 1–41, 2018.

[92] H. Jin, X. Dai, and J. Xiao, "Towards a novel architecture for enabling interoperability amongst multiple blockchains," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2018, pp. 1203–1211.

[93] B. Pillai, K. Biswas, Z. Hóu, and V. Muthukkumarasamy, "Burn-to-claim: An asset transfer protocol for blockchain interoperability," *Computer Networks*, vol. 200, p. 108 495, 2021.

[94] H. Su, B. Guo, J. Y. Lu, and X. Suo, "Cross-chain exchange by transaction dependence with conditional transaction method," *Soft Computing*, pp. 1–16, 2022.

[95] M. Sigwart, P. Frauenthaler, C. Spanring, M. Sober, and S. Schulte, "Decentralized cross-blockchain asset transfers," in *2021 Third International Conference on Blockchain Computing and Applications (BCCA)*, Tartu, Estonia: IEEE, 2021, pp. 34–41.

[96] L. Besançon, C. F. Da Silva, and P. Ghodous, "Towards blockchain interoperability: Improving video games data exchange," in *2019 IEEE international conference on blockchain and cryptocurrency (ICBC)*, IEEE, 2019, pp. 81–85.

[97] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, pp. 201–224, 2014.

[98] R. Zarick, B. Pellegrino, and C. Banister, "Layerzero: Trustless omnichain interoperability protocol," *arXiv preprint arXiv:2110.13871*, 2021.

[99] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "Zkbridge: Trustless cross-chain bridges made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3003–3017.

[100] Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, and Y.-C. Hu, "Hyperservice: Interoperability and programmability across heterogeneous blockchains," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 549–566.

[101] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of trust: A decentralized blockchain-based authentication system for iot," *Computers & Security*, vol. 78, pp. 126–142, 2018.

[102] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Lsb: A lightweight scalable blockchain for iot security and anonymity," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 180–197, 2019.

[103] B. Tang, H. Kang, J. Fan, Q. Li, and R. Sandhu, "Iot passport: A blockchain-based trust framework for collaborative internet-of-things," in *Proceedings of the 24th ACM symposium on access control models and technologies*, 2019, pp. 83–92.

[104] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 762–771, 2018.

[105] Y. Mirsky, T. Golomb, and Y. Elovici, "Lightweight collaborative anomaly detection for the iot using blockchain," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 75–97, 2020.

[106] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2019. DOI: 10.1109/JIOT.2018.2836144.

[107] C. H. Lau, K.-H. Y. Alan, and F. Yan, "Blockchain-based authentication in iot networks," in *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, IEEE, 2018, pp. 1–8.

[108] R. A. Michelin, A. Dorri, M. Steger, R. C. Lunardi, S. S. Kanhere, R. Jurdak, and A. F. Zorzo, "Speedychain: A framework for decoupling data from blockchain for smart cities," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018, pp. 145–154.

[109] J. Gao, K. O.-B. O. Agyekum, E. B. Sifah, K. N. Acheampong, Q. Xia, X. Du, M. Guizani, and H. Xia, "A blockchain-sdn-enabled internet of vehicles environment for fog computing and 5g networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4278–4291, 2019.

[110] L. Zhang, M. Luo, J. Li, M. H. Au, K.-K. R. Choo, T. Chen, and S. Tian, "Blockchain based secure data sharing system for internet of vehicles: A position paper," *Vehicular Communications*, vol. 16, pp. 85–93, 2019.

[111] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[112] D. Zhang, F. R. Yu, and R. Yang, "Blockchain-based distributed software-defined vehicular networks: A dueling deep {q}-learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1086–1100, 2019.

[113] X. Huang, C. Xu, P. Wang, and H. Liu, "Lnsc: A security model for electric vehicle and charging pile management based on blockchain ecosystem," *IEEE access*, vol. 6, pp. 13 565–13 574, 2018.

[114] S. Kudva, S. Badsha, S. Sengupta, I. Khalil, and A. Zomaya, "Towards secure and practical consensus for blockchain based vanet," *Information Sciences*, vol. 545, pp. 170–187, 2021.

[115] X. Feng, Q. Shi, Q. Xie, and L. Liu, "An efficient privacy-preserving authentication model based on blockchain for vanets," *Journal of Systems Architecture*, vol. 117, p. 102 158, 2021.

[116] Q. Mei, H. Xiong, Y. Zhao, and K.-H. Yeh, "Toward blockchain-enabled iov with edge computing: Efficient and privacy-preserving vehicular communication and dynamic updating," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, IEEE, 2021, pp. 1–8.

[117] S. Abbes and S. Rekhis, "A blockchain-based solution for reputation management in iov," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, IEEE, 2021, pp. 1129–1134.

[118] G. Liang, S. R. Weller, F. Luo, J. Zhao, and Z. Y. Dong, "Distributed blockchain-based data protection framework for modern power systems against cyber attacks," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 3162–3173, 2018.

[119] A. Jindal, G. S. Aujla, N. Kumar, and M. Villari, "Guardian: Blockchain-based secure demand response management in smart grid system," *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 613–624, 2019.

[120] J. Wang, L. Wu, K.-K. R. Choo, and D. He, "Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1984–1992, 2019.

[121] T. Wang, J. Guo, S. Ai, and J. Cao, "Rbt: A distributed reputation system for blockchain-based peer-to-peer energy trading with fairness consideration," *Applied Energy*, vol. 295, p. 117 056, 2021.

[122] Z. Guan, X. Lu, W. Yang, L. Wu, N. Wang, and Z. Zhang, "Achieving efficient and privacy-preserving energy trading based on blockchain and abe in smart grid," *Journal of Parallel and Distributed Computing*, vol. 147, pp. 34–45, 2021.

[123] J. Yang, A. Paudel, and H. B. Gooi, "Compensation for power loss by a proof-of-stake consortium blockchain microgrid," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3253–3262, 2020.

[124] Y. Cao, X. Ren, C. Qiu, X. Wang, H. Yao, and F. R. Yu, "A multi-agent reinforcement learning approach for blockchain-based electricity trading system," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6. DOI: 10.1109/GLOBECOM46510.2021.9685510.

[125] J. Yang, A. Paudel, J. Dai, and H. B. Gooi, "A mining-rewarding mechanism for peer-to-peer energy trading blockchain," in *2021 IEEE PES Innovative Smart Grid Technologies-Asia (ISGT Asia)*, IEEE, 2021, pp. 1–5.

[126] S. Karumba, S. S. Kanhere, R. Jurdak, and S. Sethuvenkatraman, "Harb: A hypergraph-based adaptive consortium blockchain for decentralized energy trading," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 216–14 227, 2022. DOI: 10.1109/JIOT.2020.3022045.

[127] T. AlSkaif, J. L. Crespo-Vazquez, M. Sekuloski, G. van Leeuwen, and J. P. S. Catalão, "Blockchain-based fully peer-to-peer energy trading strategies for residential energy systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 231–241, 2022. DOI: 10.1109/TII.2021.3077008.

[128] J. Ping, Z. Yan, and S. Chen, "A privacy-preserving blockchain-based method to optimize energy trading," *IEEE Transactions on Smart Grid*, vol. 14, no. 2, pp. 1148–1157, 2023.

[129] K. Christidis, D. Sikeridis, Y. Wang, and M. Devetsikiotis, "A framework for designing and evaluating realistic blockchain-based local energy markets," *Applied Energy*, vol. 281, p. 115 963, 2021.

[130] J. Guo, X. Ding, and W. Wu, "An architecture for distributed energies trading in byzantine-based blockchains," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 2, pp. 1216–1230, 2022.

[131] T. D. Nguyen and M. T. Thai, "A blockchain-based iterative double auction protocol using multi-party state channels," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 2, pp. 1–22, 2021.

[132] Z. Chen, W. Ding, Y. Xu, M. Tian, and H. Zhong, "Fair auctioning and trading framework for cloud virtual machines based on blockchain," *Computer Communications*, vol. 171, pp. 89–98, 2021.

[133] S. Singla, A. Dua, N. Kumar, and S. Tanwar, "Blockchain-based efficient energy trading scheme for smart-grid systems," in *2020 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2020, pp. 1–6.

[134] X. Ma, D. Xu, and K. Wolter, "Blockchain-enabled feedback-based combinatorial double auction for cloud markets," *Future Generation Computer Systems*, vol. 127, pp. 225–239, 2022.

[135] Y. Li, J. Wang, and H. Zhang, "A survey of state-of-the-art sharding blockchains: Models, components, and attack surfaces," *Journal of Network and Computer Applications*, p. 103 686, 2023.

[136] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1085–1100.

[137] E. Roth, D. Noble, B. H. Falk, and A. Haeberlen, "Honeycrisp: Large-scale differentially private aggregation without a trusted core (extended version)," Technical Report MS-CIS-19-03, University of Pennsylvania, Tech. Rep., 2019.

[138] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 17–30.

[139] S. K. Set and G. S. Park, "Service-aware dynamic sharding approach for scalable blockchain," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2954–2969, 2023. DOI: 10.1109/TSC.2022.3231619.

[140] I. Grigg, "Eos-an introduction," *White paper. https://whitepaperdatabase. com/eos-whitepaper*, 2017.

[141] A. Yakovenko, "Solana: A new architecture for a high performance blockchain v0. 8.13," *Whitepaper*, 2018.

[142] D. Yaga, P. Mell, N. Roby, and K. Scarfone, *Blockchain technology overview*, https://nvlpubs.nist.gov/nistpubs/ir/2018/nist.ir.8202.pdf, 2018.

[143] E. Fujisaki and K. Suzuki, "Traceable ring signature," in *International Workshop on Public Key Cryptography*, Berlin, Heidelberg: Springer, 2007, pp. 181–200.

[144] B. C. Inc., *An open financial system on bitcoin (liquid network)*, https://blockstream.com/liquid/, [Online; accessed August-2022], 2022.

[145] B. Narwal and A. K. Mohapatra, "A survey on security and authentication in wireless body area networks," *Journal of Systems Architecture*, vol. 113, p. 101 883, 2021.

[146] Q. Lyu, Y. Qi, X. Zhang, H. Liu, Q. Wang, and N. Zheng, "Sbac: A secure blockchain-based access control framework for information-centric networking," *Journal of Network and Computer Applications*, vol. 149, p. 102 444, 2020.

[147] S. Rasool, A. Saleem, M. Iqbal, T. Dagiuklas, S. Mumtaz, and Z. ul Qayyum, "Docschain: Blockchain-based iot solution for verification of degree documents," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 3, pp. 827–837, 2020.

[148] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, and Y. Yang, "A survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–32, 2020.

[149] T. Alladi, V. Chamola, N. Sahu, V. Venkatesh, A. Goyal, and M. Guizani, "A comprehensive survey on the applications of blockchain for securing vehicular networks," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1212–1239, 2022.

[150] Y. Wang, Y. Ding, Q. Wu, Y. Wei, B. Qin, and H. Wang, "Privacy-preserving cloud-based road condition monitoring with source authentication in vanets," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1779–1790, 2018.

[151] A. P. Sarr, P. Elbaz-Vincent, and J.-C. Bajard, "A new security model for authenticated key agreement," in *International Conference on Security and Cryptography for Networks*, Springer, 2010, pp. 219–234.

[152] I. Yaqoob, I. Ahmad, E. Ahmed, A. Gani, M. Imran, and N. Guizani, "Overcoming the key challenges to establishing vehicular communication: Is sdn the answer?" *IEEE Communications Magazine*, vol. 55, no. 7, pp. 128–134, 2017.

[153] G. S. Aujla, M. Singh, A. Bose, N. Kumar, G. Han, and R. Buyya, "Blocksdn: Blockchain-as-a-service for software defined networking in smart city applications," *IEEE Network*, vol. 34, no. 2, pp. 83–91, 2020.

[154] L. Xie, Y. Ding, H. Yang, and X. Wang, "Blockchain-based secure and trustworthy internet of things in sdn-enabled 5g-vanets," *IEEE Access*, vol. 7, pp. 56 656–56 666, 2019.

[155] A. Rahman, M. J. Islam, A. Montieri, M. K. Nasir, M. M. Reza, S. S. Band, A. Pescape, M. Hasan, M. Sookhak, and A. Mosavi, "Smartblock-sdn: An optimized blockchain-sdn framework for resource management in iot," *IEEE Access*, vol. 9, pp. 28 361–28 376, 2021.

[156] Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu, "A privacy-preserving trust model based on blockchain for vanets," *IEEE Access*, vol. 6, pp. 45 655–45 664, 2018.

[157] Y. Liu, Y. Wang, and G. Chang, "Efficient privacy-preserving dual authentication and key agreement scheme for secure v2v communications in an iov paradigm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2740–2749, 2017.

[158] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2906–2920, 2019.

[159] T. Coffey and P. Saidha, "Non-repudiation with mandatory proof of receipt," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 1, pp. 6–17, 1996.

[160] M. E. Peck and D. Wagman, "Energy trading for fun and profit buy your neighbor's rooftop solar power or sell your own-it'll all be on a blockchain," *IEEE Spectrum*, vol. 54, no. 10, pp. 56–61, 2017. DOI: 10.1109/MSPEC.2017.8048842.

[161] M. H. Cintuglu, O. A. Mohammed, K. Akkaya, and A. S. Uluagac, "A survey on smart grid cyber-physical system testbeds," *IEEE Communications Surveys Tutorials*, vol. 19, pp. 446–464, 2017. DOI: 10.1109/COMST.2016.2627399.

[162] Y. Chen, W. Wei, H. Wang, Q. Zhou, and J. P. S. Catalão, "An energy sharing mechanism achieving the same flexibility as centralized dispatch," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3379–3389, 2021. DOI: 10.1109/TSG.2021.3060380.

[163] M. H. Ullah and J.-D. Park, "Distributed energy trading in smart grid over directed communication network," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3669–3672, 2021. DOI: 10.1109/TSG.2021.3067172.

[164] J. Guerrero, D. Gebbran, S. Mhanna, A. C. Chapman, and G. Verbič, "Towards a transactive energy system for integration of distributed energy resources: Home energy management, distributed optimal power flow, and peer-to-peer energy trading," *Renewable and Sustainable Energy Reviews*, vol. 132, p. 110 000, 2020.

[165] S. H. Alsamhi, F. A. Almalki, F. Afghah, A. Hawbani, A. V. Shvetsov, B. Lee, and H. Song, "Drones' edge intelligence over smart environments in b5g: Blockchain and federated learning synergy," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 295–312, 2022. DOI: 10.1109/TGCN.2021.3132561.

[166] T. Ha, D. Lee, C. Lee, and S. Cho, "Vcg auction mechanism based on block chain in smart grid," in *2021 International Conference on Information Networking (ICOIN)*, IEEE, 2021, pp. 465–468.

[167] H. T. Doan, J. Cho, and D. Kim, "Peer-to-peer energy trading in smart grid through blockchain: A double auction-based game theoretic approach," *Ieee Access*, vol. 9, pp. 49 206–49 218, 2021.

[168] D. Li, Q. Yang, W. Yu, D. An, Y. Zhang, and W. Zhao, "Towards differential privacy-based online double auction for smart grid," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 971–986, 2019.

[169] X. Hao, W. Ren, K.-K. R. Choo, and N. N. Xiong, "A self-trading and authenticated roaming scheme based on blockchain for smart grids," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4097–4106, 2022.

[170] S. Li, K. Xue, D. S. Wei, H. Yue, N. Yu, and P. Hong, "Secgrid: A secure and efficient sgx-enabled smart grid system with rich functionalities," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1318–1330, 2019.

[171] Q. Yang and H. Wang, "Privacy-preserving transactive energy management for iot-aided smart homes via blockchain," *IEEE Internet of Things Journal*, 2021.

[172] A. P. Ozisik and B. N. Levine, "An explanation of nakamoto's analysis of double-spend attacks," *Arxiv*, 2017. DOI: 10.48550/arXiv.1701.03977. eprint: https://arxiv.org/abs/1701.03977.

[173] M. El-Shehawey, "On the gambler's ruin problem for a finite markov chain," *Statistics & Probability Letters*, vol. 79, no. 14, pp. 1590–1595, 2009.

[174] H. Bao, B. Ren, B. Li, and Q. Kong, "Bbnp: A blockchain-based novel paradigm for fair and secure smart grid communications," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 12 984–12 996, 2022. DOI: 10.1109/JIOT.2021.3107301.

[175] I. Dimobi, M. Pipattanasomporn, and S. Rahman, "A transactive grid with microgrids using blockchain for the energy internet," in *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, IEEE, 2020, pp. 1–5.

[176] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," *GitHub: San Francisco, CA, USA*, vol. 4, p. 220, 2016.

[177] A. Al-Mamun and D. Zhao, "Echain: Securing electronic device provenance through privacy-preserving consortium blockchains," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, 2023, pp. 77–83. DOI: 10.1109/DSN-S58398.2023.00028.

[178] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, Springer, 2001, pp. 552–565.

[179] R. L. Rivest, L. Adleman, M. L. Dertouzos, *et al.*, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[180] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12, Cambridge, Massachusetts: Association for Computing Machinery, 2012, pp. 326–349, ISBN: 9781450311151. DOI: 10.1145/2090236.2090263.

[181] N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer, *The hunting of the snark*, https://eprint.iacr.org/2014/580, 2014. [Online]. Available: https://eprint.iacr.org/2014/580.