Blockchain-based Advanced Information Infrastructure and its Applications in Demand-Side Energy Systems

TENG YU

B.E., M.E. in Electrical Engineering

Supervisor: Dr Fengji Luo
Associate Supervisor: Prof. Gianluca Ranzi

A thesis submitted in fulfilment of the requirements for
the degree of Doctor of Philosophy

School of Civil Engineering
Faculty of Engineering
The University of Sydney
Australia

2026

Abstract

The energy sector is moving from centralised one-way power systems to decentralised bidirectional smart grids driven by distributed energy resources (DERs). This shift requires infrastructure for secure coordination, verifiable markets, and prosumer data privacy. Existing blockchain systems have limited scalability, security, and interoperability, restricting high-frequency smart-grid applications. This thesis designs blockchain infrastructure for peer-to-peer (P2P) energy trading and power load forecasting (PLF).

It has three theoretical innovation points. First, a dual-blockchain architecture with an Improved Optimistic Rollup (IOR) improves vertical throughput by offloading heavy computation from a primary to a secondary blockchain. Second, a Transaction Batch Generation (TBG) protocol for leaderless Byzantine Fault Tolerance (LBFT) consensus improves horizontal throughput by letting every node broadcast blocks, avoiding redundant transaction rebroadcast, and reducing transaction censorship to nearly zero. Third, a Blockchain-of-Blockchains (BoB) architecture, with Cross-Chain Token Exchange (CCTE) and Cross-Chain Data Interoperability (CCDI), supports asset and data transfer across blockchains with lower latency and memory overhead. All the three innovative points improve the state-of-the-art blockchains' performance, while also enhancing their security.

The infrastructure is applied to two demand-side problems. For P2P market clearing, it is combined with Trusted Execution Environments (TEEs), D-TASK, and TEAR-DO to address the privacy-robustness-latency trilemma: prosumer data remain private, distributed optimisation preserves optimal convergence, and clearing time is reduced by an order of magnitude over state-of-the-art methods. For PLF, CTP-FL combines the infrastructure with message coding and commitment-based dual consensus to preserve local-model privacy, tolerate Byzantine faults at client and server sides, and keep communication latency below local model training time. This yields secure,

scalable, interoperable blockchain infrastructure for real-time demand-side energy applications.

Acknowledgements

I would like to express my deepest gratitude to my principal supervisor, Dr. Fengji Luo, and my associate supervisor, Professor Gianluca Ranzi, for their invaluable guidance and unwavering support throughout my PhD journey.

I am especially indebted to Dr. Fengji Luo, whose rigorous academic standards and personal dedication have profoundly shaped my development. I will never forget the countless hours he spent painstakingly refining my manuscripts, line by line and word by word, even when my writing was in its clumsiest early stages. His patience during those sessions did more than just improve my papers; his direct and insightful mentorship instilled in me the mindset of a true researcher.

My deepest thanks also go to my parents, my beloved wife, Yanan Peng, my sister Bin Yu and my uncles. They have been my emotional anchor and sanctuary throughout these four years. To Yanan: thank you for your unconditional belief in me and for bearing the weight of this long journey by my side.

I am also sincerely grateful to my colleagues Yu He, Lanyi Zhang, Zhichen Ye, Bohan Zhang, Jianhen Lan, and other schoolmates. Special thanks go to Yu He, and I will always cherish the memory of us untangling tricky mathematical knots while walking by the seaside, where the sea breeze and our discussions sparked some of my research passion.

I extend my heartfelt thanks to my friends Feng Xiong, Junhui Sun, Jiewei, Qiubo Han, Yuhan Wang, Chong Chen and also Hongzhi Xing, Weiwen Nie, Yifan Jiang, Wei Zhu and others. Your generosity – both spiritual and financial – served as a vital lifeline. Your support sustained me through the most challenging times, allowing me to cross the finish line of this PhD “marathon”.

I would also like to thank the University of Sydney for providing access to the research facilities, computational infrastructure, and academic environment that made this work possible.

List of Publications

Peer-reviewed journal articles:

(1) **T. Yu**, F. Luo, G. Ranzi, and J. Wu, "Secure and efficient data interoperability protocols for multi-blockchains systems," *IEEE Transactions on Information Forensics and Security*, pp. 11401–11416, 2025.

(2) **T. Yu**, F. Luo, C. Pu, Z. Zhao and G. Ranzi, "Dual-Blockchain-Based P2P Energy Trading System With an Improved Optimistic Rollup Mechanism," *IET Smart Grid*, pp. 246–259, 2022. (**Featured article**)

(3) **T. Yu**, F. Luo, and B. Yu, "TBG: A Batch Generation Mechanism for Leaderless Byzantine Fault Tolerance Protocols," *IEEE Transactions on Information Forensics and Security*, 2026 (accepted).

Book chapter:

(4) **T. Yu**, F. Luo, Q. Wu and G. Ranzi, "Blockchain in smart grids: a review of recent developments." in *Emerging Smart Technologies for Critical Infrastructure*. Springer, 2023, pp. 23–59.

Full papers in conferences proceedings:

(5) **T. Yu**, F. Luo, and G. Ranzi, "PCCAE: A protocol for multi-party asset exchange among blockchains," *in Proc. ACM SIGCOMM Workshop Zero Trust Archit. Next Gener. Commun*., 2024, Sydney, Australia.

Submitted journal articles:

(6) **T. Yu**, F. Luo, and G. Ranzi, "Trustworthy and Efficient Framework for Solving Trilemma in Distributed Optimisation-based Peer-to-Peer Energy Market Clearance," *IEEE Transactions on Smart Grid*, 2025, under review.

(7) **T. Yu**, F. Luo, Y. He, Z. Ye and G. Ranzi, “Efficient and Byzantine Tolerant Federated Learning System for Power Load Forecasting,” *IEEE Transactions on Information Forensics and Security*, 2025, under review.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Teng Yu (Signature)

May 2026

Authorship Attribution Statement

This thesis contains the following materials:

(1) **T. Yu**, F. Luo, C. Pu, Z. Zhao and G. Ranzi "Dual-Blockchain-Based P2P Energy Trading System With an Improved Optimistic Rollup Mechanism," *IET Smart Grid*, pp. 246–259, 2022. This publication is Chapter 3 of this thesis.

(2) **T. Yu**, F. Luo, and B. Yu, "TBG: A Batch Generation Mechanism for Leaderless Byzantine Fault Tolerance Protocols," *IEEE Transactions on Information Forensics and Security*, 2026, accepted. This manuscript is Chapter 4 of this thesis.

(3) **T. Yu**, F. Luo, G. Ranzi, and J. Wu, "Secure and efficient data interoperability protocols for multi-blockchains systems," *IEEE Transactions on Information Forensics and Security*, pp. 11401–11416, 2025. This publication is Chapter 5 of this thesis.

(4) **T. Yu**, F. Luo, and G. Ranzi, "Trustworthy and Efficient Framework for Solving Trilemma in Distributed Optimisation-based Peer-to-Peer Energy Market Clearance," *IEEE Transactions on Smart Grid*, 2025, under review. This manuscript is Chapter 6 of this thesis.

(5) **T. Yu**, F. Luo, Y. He, Z. Ye and G. Ranzi, "Efficient and Byzantine Tolerant Federated Learning System for Power Load Forecasting," *IEEE Transactions on Information Forensics and Security*, 2025, under review. This manuscript is Chapter 7 of this thesis.

Chapters 4 and 5 are based on author accepted manuscript versions of articles accepted or published in IEEE Transactions on Information Forensics and Security. The final published IEEE versions are not reproduced in this thesis. IEEE copyright notices are provided at the beginning of the relevant chapters, with reprinted figures and tables, and in the Bibliography.

IEEE material in this thesis is included for thesis and dissertation use. IEEE does not endorse The University of Sydney's products or services. Any reprinting or republication of IEEE material for advertising, promotional use, resale, redistribution, or new collective works requires a separate licence from IEEE RightsLink.

In addition to the statements above, in cases where I am not the corresponding author of the published item, permission to include the published material has been granted by the corresponding author.

Teng Yu (Signature)

May 2026

As a supervisor for the candidate upon whom this thesis is based, I can confirm that the authorship attribution statements above are correct.

Fengji Luo, (Signature)

May 2026

List of Abbreviations

ABA	Asynchronous binary agreement
ACS	Asynchronous common subset
AMM	Automated market maker
ATs	Agreement transactions
BC	Blockchain
BFT	Byzantine fault tolerance
BoB	Blockchain of blockchains
CCA	Corruption-chasing attack
CCDI	Cross-chain data interoperability
CCTE	Cross-chain token exchange
DAG	Directed acyclic graph
DDoS	Distributed denial of service
DERs	Distributed energy resources
DO	Distributed optimisation
EC	Erasure coding
ESSA	Epoch-skew scheduling attack
FBSA	Flooding-based batch saturation attack
FL	Federated learning
HE	Homomorphic encryption
HSM	Hash space method
HTLCs	Hash timelock contracts
IOR	Improved optimistic rollup
LBFT	Leaderless Byzantine fault tolerance
LP	Liquidity pool
MBS	Multi-blockchain system
MED	Message encryption and decryption

MPC	Multi-party computation
MSM	Multi-scalar multiplication
MVBA	Multi-value Byzantine agreement
OR	Optimistic rollup
P2P	Peer-to-peer
PBPC	Position-binding polynomial commitment
PLF	Power load forecasting
PTDFs	Power transfer distribution factors
RA	Remote attestation
RB	Redundant broadcasts
RBC	Reliable broadcast
SRS	Structured reference string
STs	Settlement transactions
TBG	Transaction batch generation
TC	Transaction censorship
TCRs	Transaction computation results
TEE	Trusted execution environment
TMA	Trimmed mean aggregation
TS	Threshold signature
ZKR	Zero knowledge-based rollups

Artificial Intelligence

During the preparation of the thesis the author used ChatGPT (see webpage at https://chatgpt.com) for the purposes of text enhancement. This includes paraphrasing, sentence-structure improvement, and spelling correction. The author confirms that where text was modified by generative AI, the content was reviewed for possible errors, inaccuracies, and bias. The author takes full responsibility for the submitted thesis and ensures the work is their own and has used generative AI within the parameters of use (refer to the University of Sydney generative AI guide for researchers).

Teng Yu (Signature)

May 2026

Australian Government Support

I acknowledge the financial support received through the University of Sydney Postgraduate Research Support Scheme 2024 and Postgraduate Research Support Scheme 2025.

List of Figures

List of Tables

Contents

CHAPTER 1
Introduction

In this thesis, an advanced blockchain-based information infrastructure is devised, which acts as the secure, trustworthy and high-performance information system for applications in demand side energy systems.

1.1 Motivation and Background

This chapter presents the background, motivations, and the proposed approach to build the information infrastructure.

1.1.1 Demand-Side Energy Systems and Their Information Infrastructure

The global energy sector is undergoing a profound transition driven by the large-scale deployment of distributed energy resources (DERs), such as rooftop photovoltaics, battery energy storage systems, electric vehicles, and responsive loads [1]. Instead of being passive consumers, many end-users have become prosumers who both generate and consume electricity. This shift is reshaping traditional power systems into demand-side-driven energy systems, where a significant portion of decision-making operations moves from central utilities to distributed end-users.

In such demand-side-driven energy systems, the physical layer of power networks must be complemented by an equally sophisticated information infrastructure [2]. This information infrastructure has to support secure data collection, enable trustworthy coordination among heterogeneous prosumers and grid managers, provide verifiable execution of market mechanisms, and ensure privacy preservation for prosumers' sensitive energy data. Without a reliable information layer, the potential of demand-side energy applications cannot be securely and reliably realised, making energy system remain fragile, siloed, or overly centralised [3].

1.1.2 Demand-Side Applications: P2P Energy Trading and Power Load Forecasting

Currently, massive DERs are penetrated into power systems, which thus spawns two demand-side applications, i.e., Peer-to-Peer (P2P) energy trading and power load forecasting (PLF).

First, P2P energy trading represents a decentralised market paradigm enabling prosumers to trade surplus energy directly. Underpinned by a digital platform serving as the transactional infrastructure, this model facilitates market clearing by mathematically matching supply and demand and executing financial settlements. All the resulting matching and settlement transactions are finally recorded in the digital platform. However, as the ecosystem scales to include more heterogeneous prosumers and diverse trading platforms, the volume and complexity of transactions increase. The supporting digital platform must now process a high frequency of varied bids and offers, and manage a correspondingly complex settlement of multiple asset types. Given the inevitable complexity, the platform must rigorously enforce computational integrity (i.e., ensuring market clearing algorithms are executed exactly as coded) and operational fairness (i.e., effectively immunising the system against malicious manipulation such as false data injection or protocol deviations [4]). How to ensure both requirements while maintaining high executional performance under high transaction volume is worth of investigation.

Second, accurate power load forecasting – the prediction of future energy (in most case, electricity) demand – is fundamental to demand-side management, enabling critical applications such as dynamic energy pricing and local energy balancing. However, with the increasing penetration of DERs, achieving this accuracy now requires the collection and analysis of high-resolution data distributed across numerous prosumers and devices. The conventional approach to harness this data is centralised machine learning, where all data is collected and processed on a single server. This method, however, raises significant privacy concerns for prosumers and creates a single

point of failure at the central server [5]. More critically, as data volumes expand, centralised architectures face prohibitive computational and communication bottlenecks, precluding the high-frequency machine learning model updates necessary to capture the stochastic dynamics of DERs. To date, existing literature fails to simultaneously reconcile three conflicting objectives: privacy preservation of raw energy training data, Byzantine robustness against both data poisoning and trainer compromise, and training efficiency scalable to high-dimensional datasets.

These two application areas – P2P energy trading and power load forecasting – form the core demand-side application domains that this thesis targets.

1.1.3 Blockchain as an Information Infrastructure: Promises and Limitations

Blockchain (BC) technology has emerged as a promising candidate for building such an information infrastructure in energy systems [3]. Blockchain technology has emerged as a promising foundation for this required infrastructure. Its core attributes directly address the needs of a decentralised energy system:

- **Consensus protocols:** consensus protocols are pre-specific operational and communicational rules that allow all participants to agree on transaction validity without a central authority.
- **Immutable, tamper-evident recording:** Every transaction is cryptographically sealed into a permanent, shared ledger, creating a transparent and auditable history for all data flows and market actions.
- **Self-executing smart contracts:** Blockchain provides smart contracts functionality which ensures that the deployed codes are executed correctly via redundant computing.

Consequently, a significant body of research now explores blockchain-based solutions for P2P energy trading and power load forecasting [6].

Despite these advantages, directly deploying existing blockchains in energy systems remains problematic. First, scalability is limited: traditional permissionless

blockchains process only tens of transactions per second. This capacity fails to accommodate the massive volume of concurrent micro-transactions required for widespread P2P energy trading [7]. Second, the blockchain trilemma highlights an inherent tension between scalability, security, and decentralisation. Naïve attempts to increase throughput often sacrifice decentralisation or security guarantees, which is unacceptable for safety-critical energy infrastructures [8]. Third, most existing designs are single-chain and application-specific. They are often engineered for a single community, microgrid, or service, and struggle to support large-scale, cross-region application scenarios such as nation-wide P2P energy trading, market clearing across multiple interconnected microgrids, or hierarchical coordination between local and regional markets [9]. In such settings, the number of participants, diversity of stakeholders, and heterogeneity of operational requirements quickly exceed what a monolithic blockchain can sustain.

Moreover, a blockchain ledger by itself mainly provides trustworthy state replication and traceable data storage. Yet demand-side energy system applications require much more from the information infrastructure [10]. For example, P2P market clearing must protect prosumers' private energy information while still enabling real-time clearing of energy trading pairs [11]; power load forecasting increasingly relies on fine-grained, reliable energy consumption and generation data from individual households or devices. As a result, the infrastructure must also provide the confidentiality of prosumers' energy profiles, and at the same time defend against Byzantine prosumers that inject fabricated or manipulated energy-related data to distort operational accuracy [12]. Similar patterns also arise in other demand-side services, where energy data confidentiality, low operational latency, and robustness to malicious behaviour are all first-class requirements of demand-side energy applications [13].

These additional requirements effectively push the blockchain role beyond a passive ledger towards an active information infrastructure that can efficiently interoperate with cryptographic protocols [14], trusted execution environments [15], and distributed learning schemes [16]. In other words, future demand-side energy

systems do not merely need a blockchain for immutable logging; they need an advanced blockchain-based information infrastructure that simultaneously (i) overcomes core performance limitations of current blockchain systems (ii) natively preserve privacy of sensitive energy data, and (iii) Byzantine-resilience at the prosumer side. Also, the infrastructure needs to be compatible to state-of-the-art cryptographic protocols for broader security purposes, e.g., transaction censorship resistance. How to architect such an infrastructure is the central question of this thesis.

1.2 Problem Statement and Objective

To ensure the three requirements (i)- (iii) specified in the end of Section 1.1.3, this thesis is organised around the following two design goals:

Design goal 1 – Infrastructure capability. The primary objective is to devise a high-performance blockchain infrastructure characterised by a heterogeneous protocol stack. By integrating improved Layer-2 scaling mechanisms, advanced consensus algorithms, and secure cross-chain interoperability, this advanced blockchain architecture is designed to circumvent the fundamental scalability and efficiency bottlenecks of conventional monolithic blockchains.

Design goal 2 – Application support. The second goal is to demonstrate that the resulting blockchain-based information infrastructure can naturally host or additionally coordinate more features that the traditional blockchain-based information infrastructure cannot provide, e.g., the efficient, privacy-preserving computation techniques and state-of-the-art cryptographic protocols. All these provided new features can satisfy the requirements of the two considered demand-side energy applications, i.e., P2P energy trading and power load forecasting.

In summary, Design goal 1 concerns how the underlying information infrastructure is constructed and improved. Design goal 2 concerns how this infrastructure is instantiated in concrete applications so that these applications benefit from the security and performance of the infrastructure. Together, these two goals delineate the scope of

the thesis and form the conceptual bridge between the theory-oriented contributions in Chapters 3-5 and the application-oriented contributions in Chapters 6-7.

1.3 Summary of Contributions

Accordingly, to achieve the two design goals outlined in Section 1.2, this thesis adopts a two-stage methodology:

1. **Infrastructure stage (Chapters 3–5).** We develop an advanced blockchain infrastructure by integrating the proposed layer-2 mechanism, robust consensus protocol, and efficient cross-chain protocols, that collectively realise "Design Goal 1". This stage provides the *theory-level* innovations of this thesis.
2. **Application stage (Chapters 6–7).** We instantiate the infrastructure in two representative demand-side energy applications – P2P energy market clearing and power load forecasting – and design additional mechanisms needed to fully realise "Design Goal 2". This stage provides the *application-level* innovations of this thesis.

1.3.1 Infrastructure-Focused Research Questions and Answers (Chapters 3–5)

The theoretical contributions are organised around the following three research questions (RQs) and their corresponding answers.

1 **RQ1 – Layer-2 mechanism.**

How can we improve the state-of-the-art blockchain systems' throughput via a layer-2 design, that jointly addresses (i) long dispute-resolution (challenge) periods that current layer-2 schemes require, and (ii) the centralised risk that layer-2 executors collude with corresponding validators?

Answer: Chapter 3 tackles this question by introducing a dual-blockchain architecture equipped with an Improved Optimistic Rollup (IOR) mechanism. The design separates operational responsibilities vertically: a primary blockchain (layer-1) acts as a lightweight ledger that records transaction data and transaction

execution results, while a secondary blockchain (layer-2) serves as a computation layer where a small set of executor nodes run the transaction execution tasks to output the results. On top of this dual-chain structure, the IOR mechanism refines conventional Optimistic Rollup (OR) via two dimensions:

First, an "executor-validator-delegated validators" pipeline replaces long open challenge periods required in the existing OR mechanisms. For each batch of transactions, an executor on the secondary chain computes the results of these transactions, where a transaction can be treated as a computational task; a validator node immediately re-processes the same transactions; and a group of delegated validators independently spot-checks the validity of the results produced by the executor and the validator. This design makes collusion between the executor and the validator economically unattractive while eliminating the need of the long open challenge periods existing in the OR schemes.

Second, because the primary chain only runs simple data recording contracts, while heavy computation is offloaded to the secondary chain, prototype experiments on P2P energy settlement show that the IOR-based dual-chain structure provides higher throughput and lower on-chain cost than a single-chain baseline. In summary, the IOR-based dual-chain design provides a vertically scalable layer to the advanced blockchain-based information infrastructure.

2 **RQ2 – Leaderless consensus and batch generation.**

How can we achieve a highly efficient consensus protocol while eliminating its security vulnerabilities, under a realistic corruption model where an adversary can continually corrupt consensus nodes?

Answer: Chapter 4 answers this question by designing two underlying protocols for leaderless Byzantine Fault Tolerance (LBFT) protocols. Specifically, LBFT protocols horizontally improve the throughput and latency of consensus protocols, as every consensus node in LBFT protocols, instead of a leader node in leader-based Byzantine Fault Tolerance (BFT) protocols, can broadcast blocks of transactions in

parallel. However, current LBFT protocols suffer from redundant broadcasts – a transaction is repeatedly broadcast by multiple nodes, which wastes system's bandwidth and thus reduces system's throughput. In addition, the state-of-the-art LBFT protocols also provide opportunities for adversaries to prevent some targeted transactions from being executed (called transaction censorship) under realistic corruption models, which undermines system's security.

To address transaction redundancy and censorship, this chapter proposes a τ-partition transaction protocol π_τ and a Transaction Batch Generation (TBG) protocol. π_τ can mitigate the issue of transaction censorship, while TBG then resolves the issue of transaction redundancy. By combining π_τ and TBG with a LBFT protocol, analysis and simulations show up to 21.6× higher throughput, 9.1× lower latency, and near-zero censorship compared with state-of-the-art LBFT protocols. Thus, Chapter 4 offers a horizontally scalable, secure consensus layer to the advanced blockchain-based information infrastructure.

3 **RQ3 – Cross-chain coordination.**

How can we design secure and efficient mechanisms that coordinate data and asset operations across multiple blockchains, so that large-scale, cross-region smart grid ecosystems are not constrained by a single chain?

Answer: Chapter 5 answers this question by placing the blockchains from Chapters 3-4 into a multi-blockchain architecture – a dedicated blockchain-of-blockchains (BoB) connects other external blockchains; each external blockchain keeps only its own state and independence, while the BoB stores the extra information needed to validate cross-chain operations.

Under the multi-blockchain architecture, this chapter first introduces a cross-chain token exchange (CCTE) protocol, which lets two parties exchange tokens on different blockchains atomically – either both of the parties' transfers succeed or both fail – without heavy smart contract-based operations required by state-of-the-art schemes. Then, a generic Cross-Chain Data Interoperability (CCDI) protocol is

proposed that enables arbitrary cross-chain operations (e.g. data migration, cross-region P2P markets). Together, CCTE and CCDI provide the interoperability layer to the advanced blockchain-based information infrastructure.

1.3.2 Application-Focused Research Questions and Answers (Chapters 6–7)

The application contributions are organised around the following two research questions (RQs) and their corresponding answers.

RQ4 – time-efficient, Byzantine tolerant and privacy-preserving distributed optimisation for P2P market clearing.

Given the infrastructure developed in RQ1–RQ3, how can we further design a P2P energy market clearing framework that simultaneously ensures (i) privacy of prosumers' data, (ii) optimal convergence of distributed optimisation (DO) algorithms in the presence of Byzantine blockchain nodes, and (iii) real-time market clearing?

Answer: Chapter 6 answers this question by building a TEE-assisted market-clearing framework on top of the advanced blockchain infrastructure developed from Chapters 3-5. Each blockchain node hosts a Trusted Execution Environment (TEE), i.e., a hardware enclave that can decrypt and process data without exposing the data to its hosting operator. With the support of TEEs, prosumers encrypt their bids (offers) and other local parameters and submit the ciphertexts to the blockchain; TEEs then decrypt the committed ciphertexts and jointly run the DO-based clearing algorithm. Since plaintexts are solely accessed by TEEs, the data privacy issue is resolved.

The key challenge is that a naïve "TEE+DO" design either incurs prohibitive delays due to TEE's per-iteration proofs or trusts every TEE even when some of their hosting nodes are Byzantine. The framework overcomes this by designing D-TASK (which reliably selects a set of TEEs who configure with an identical secret) and proposing TEAR-DO (which utilises the identical secret to replace TEE's original

heavy proofs with lightweight, signature-based proofs). By integrating TEEs and the two proposed strategies into the advanced blockchain system, the renewed system keeps all sensitive data inside TEEs for energy data privacy, guarantees convergence of the DO algorithm even in the presence of compromised blockchain nodes, and reduces clearing time by roughly an order of magnitude compared with a state-of-the-art method, thereby resolving the privacy-robustness-latency tension for DO-based P2P market clearing.

RQ5 – Byzantine-tolerant federated learning for power load forecasting.

How can we design a federated learning (FL) system for power load forecasting that, building on the advanced blockchain-based infrastructure, achieves three properties at once: (i) communication latency lower than local training time, (ii) strong privacy of local model parameters, and (iii) robustness to Byzantine behaviour from both clients and blockchain nodes?

Answer: Chapter 7 addresses this question by proposing CTP-FL, a federated learning paradigm for power load forecasting built on the advanced blockchain infrastructure developed in Chapters 3-5. CTP-FL's core idea is to redistribute the communication load of the central server: instead of assigning a central server or blockchain leader node to receive local models from users or broadcast a global model to users, this bandwidth overhead of the leader node is evenly distributed to every BC node via a message-coding protocol. To further ensure the security of this design, a proposed dual consensus protocol is further integrated into the advanced blockchain system. Taken together, CTP-FL cuts per-iteration communication latency of FL by over an order of magnitude compared with single-chain or server-based FL, keeps local model parameters confidential, and sustains stable forecasting accuracy even when a significant fraction of clients and blockchain nodes behave maliciously.

1.4 Thesis Outline

The remainder of this thesis is organised as follows.

Chapter 2 provides a comprehensive review of blockchain and its applications in energy systems, with a particular focus on the two applications, i.e., P2P energy trading and power load forecasting.

Chapter 3 introduces the proposed dual-blockchain architecture with an improved optimistic rollup mechanism for P2P energy settlement. This chapter simultaneously contributes to the theoretical understanding of layer-2 mechanisms and demonstrates their impact on P2P energy settlement performance.

Chapter 4 presents the protocol TBG for LBFT consensus protocols. The chapter analyses the limitations of existing leaderless consensus schemes in terms of transaction redundancy and censorship, describes the TBG design in detail, and assesses its performance improvements via theoretical analysis and simulation.

Chapter 5 develops the blockchain-of-blockchains (BoB) architecture and two cross-chain protocols: the CCTE protocol and the CCDI protocol for cross-chain token and data operations, respectively. The chapter analyses their security against two attacks and benchmarks their performance against state-of-the-art cross-chain approaches.

Chapter 6 applies the advanced blockchain-based information infrastructure to P2P energy market clearing. The market clearing system integrates TEEs and combines the blockchain-based infrastructure specified in Chapters 3-5 with the proposed two strategies, i.e., D-TASK and TEAR-DO, to achieve time-efficient, privacy-preserving, and Byzantine-tolerant market clearing. Extensive case studies demonstrate the benefits of the system in realistic P2P market scenarios.

Chapter 7 applies the blockchain-based infrastructure to federated learning-based power load forecasting. It introduces the CTP-FL paradigm, details the message coding, and dual-consensus protocols, and evaluates the system under various attack models

and practical settings. The numerical results highlight the feasibility and advantages of implementing FL for power load forecasting over the advanced blockchain system.

Chapter 8 concludes the thesis by summarising the main findings, discussing practical implications for energy system deployments, and outlining future research directions.

CHAPTER 2

Literature Review

Traditional power systems were designed for one-way power flow from large power plants to consumers, operated through centralised control [17]. In this model, system operators relied on limited data, such as aggregate power flow at substations collected every few minutes, to manage the grid. The growth of distributed energy resources, flexible loads and prosumers has changed this operating model [18]. The demand side is no longer passive; it now contains many active participants that can provide flexibility while increasing operational and information-system complexity. New entities, such as local energy communities, P2P trading platforms and commercial aggregators, have emerged to coordinate active participants that may operate under different regulations and across different geographical regions. Consequently, the information system of the power grid must evolve to match this new physical reality. This evolution requires a transition from coarse-grained energy data to fine-grained, near-real-time telemetry from active participants. The information infrastructure must also support precise coordination among active participants and new entities. These capabilities should be realised without compromising the robustness of the information system, including the privacy of detailed user energy data and resistance to malicious manipulation [19]. These requirements for coordination, transparency, privacy and security have motivated interest in decentralised, tamper-evident infrastructures, especially blockchain systems, as candidates for the information backbone of smart grids [3]. The remainder of this chapter reviews such systems and their applications to P2P energy trading and power load forecasting.

2.1 Background of Blockchain in Energy Systems

Blockchain is a distributed ledger technology in which a set of nodes maintains a replicated, append-only log of records, called blocks [20-22]. Each block contains a batch of transactions, a cryptographic hash of the previous block and auxiliary metadata [20-22]. The hash-linked block structure makes unauthorised modification detectable under standard cryptographic assumptions [20]. Consensus protocols allow blockchain nodes to agree on a single, globally ordered sequence of blocks even when some nodes are Byzantine [21]. Smart contracts are deterministic programs that encode market rules, settlement logic and access-control policies [21]. Once their pre-specified conditions are satisfied, these contracts are executed by blockchain nodes to support auditable computation [21]. Taken together, the replicated ledger, consensus mechanisms and smart contracts form a decentralised infrastructure that enables mutually distrustful parties to coordinate without relying on a single central operator [21]. Depending on who is allowed to participate in consensus, blockchains are commonly classified as permissionless, where participation is open to anyone, or permissioned, where participation is restricted to authorised nodes [23].

In energy systems, blockchain has been investigated as a building block for several demand-side applications [23-25]. Typical examples include P2P energy trading [26] and token-based incentive schemes for electric-vehicle charging [27]. Other works use the ledger primarily as a secure logging and authorisation layer to record metering and billing information, manage prosumer identities, or support data sharing among prosumers, retailers and third-party service providers [28, 29]. In these settings, blockchain-based designs are motivated by transparent transaction logging, reduced dependence on central platforms and improved robustness against Byzantine behaviour in the information infrastructure [30].

2.1.1 Security Scope and Threat Model for Smart-Grid Applications

This section clarifies the security scope used across the smart-grid chapters. The section defines which threats are addressed by the proposed blockchain-based information infrastructure and which physical-layer or operator-side controls remain outside the Thesis's scope.

This thesis focuses on cyber-layer processes that support demand-side smart-grid applications. The cyber layer denotes the modules including blockchain, the data transmission channel and the computation engines. These modules handle information originally generated smart meters or other data measuring devices. These device-generated data is considered as trustworthy and essential for computational tasks such as market clearing algorithms, model training algorithms or grid side management algorithms. Therefore, the protocols in this thesis do not model physics-layer security, such as meter-calibration bypass, relay compromise, unsafe switching or other possible emergency operator actions [9, 19]. This boundary does not imply that endpoint attacks are unimportant; it specifies that they are mitigated outside the protocols considered in this thesis.

Within this boundary, the relevant adversarial behaviours occur after the data, e.g., smart meters' measurements, are generated and further used by the participants for computational tasks of smart grid applications. These participants, like prosumers, aggregators, blockchain nodes and forecasting clients, can perform Byzantine behaviour. Here, Byzantine behaviour means arbitrary protocol (or rule) deviation by an admitted participant, including sending incorrect messages that are not the ones produced by authorised entities or devices, withholding required messages, producing inconsistent outputs, colluding with other participants and others. Privacy attacks may also occur when an admitted participant uses the visible information to infer private data. Consequently, the realistic threats considered in Chapter 3, Chapter 6 and Chapter 7 are modifying the correctly generated device data by participants, generating incorrect computational results by incorrectly executing algorithms (e.g., clearing or settlement

results), transaction censorship or withholding attacks to prevent data from receiving by corresponding receivers, and privacy leakage from sensitive energy data or model parameters. Also, the considered threats can be related to other possible Byzatnine behaviours conducted by participants who are eligible for managing, processing the device-generated data, designing and executing algorithms, and transmitting information to others. These threats are realistic for demand-side energy systems because the above participants may be operated by mutually distrustful parties.

In Chapter 3, this scope leads to a settlement-correctness threat model for peer-to-peer (P2P) energy trading. Smart meters are treated as measurement devices and sources of local evidence about a prosumer's energy production and consumption. Hence, we treat the smart meter-generated data as correct and trustworthy. The main adversarial concern is that an executor (i.e., the participant described above) may submit incorrect transaction computation results (TCRs), a validator may fail to report an inconsistency or collude with the executor.

In Chapter 6, the threat model concerns privacy leakage and Byzantine behaviour during blockchain-based P2P market clearing. The sensitive data generated by smart meters or other trustworthy devices include prosumers' bids, offers, utility or cost parameters, coupling variables and grid parameters used for distributed optimisation-based market clearing [233, 241]. If exposed, these data can reveal energy profiles, commercial preferences or grid constraint information. A Byzantine blockchain node may withhold required outputs, submit messages that fail validation, or delay distributed optimisation (DO) convergence. These behaviours can censor certain prosumers' bidding or offering transactions, generating misleading optimisation results to interrupt grid's operational decisions, and enabling attackers to profit from their Byzantine behaviours.

In Chapter 7, the threat model concerns FL-based power load forecasting (PLF). Physical load records remain at clients, while clients exchange local model parameters or coded model fragments during iterative training. Byzantine clients may submit maliciously modified model parameters, and Byzantine blockchain nodes may drop,

alter, withhold or incorrectly aggregate model parameters. These Byzantine behaviours will make the globally generated model useless for PLF.

2.2 Strategies for Improving Blockchain's Performance

With this security scope established, current blockchain platforms still fall short of the requirements of large-scale demand-side energy applications [30]. First, limited throughput and non-negligible block generation latency make it difficult to support high-frequency P2P trades or near real-time local market clearing at the scale of thousands of devices [31]. Second, existing smart-grid prototypes are typically built on isolated, single-chain platforms, often tailored to one community, microgrid or service, with weak interoperability across blockchains [32]. These limitations motivate the more detailed reviews in the following sections: layer-2 strategies for vertical scalability, more efficient consensus protocols for horizontal scalability, and cross-chain coordination schemes for secure interoperability in multi-blockchain systems [32].

2.2.1 Reviews on Layer-2 Schemes

Layer-2 protocols address blockchain scalability by offloading transaction execution from the primary blockchain (Layer-1) while inheriting its security [31]. Early solutions, such as payment channels and Plasma [33-35], pioneered this approach but faced significant limitations: channels often require users to remain online and are restricted to simple payments [34], while Plasma struggles with complex examination of the correctness of layer-2's computational results [35]. Consequently, Rollups have emerged as the dominant architecture [36]. Unlike their predecessors, rollups batch transactions and upload computational results of these transactions directly to the Layer-1 [37, 38]. The results can be verified by smart contracts deployed in Layer-1.

Among rollup variants, Optimistic Rollups (ORs) – exemplified by Arbitrum [39] – prioritise ease of implementation. They operate on a 'presumption of validity', meaning the system assumes layer-2 execution is correct by default to maximise

throughput via eliminating redundant computation on the same execution via smart contracts [38]. Consequently, Layer-1 nodes do not execute and verify transactions upfront; instead, integrity of transactions' results is enforced retroactively via a dispute mechanism [40]. This mechanism establishes a substantial 'challenge window' (e.g., a week). If a verifier detects an invalid transaction's result during this period, they submit a fraud proof to the Layer-1 to launch a dispute process. This proof triggers the Layer-1 nodes to re-execute the specific disputed transactions to definitively adjudicate the correct results. This 'reveal-then-challenge' architecture acts as an economic deterrent, ensuring honest verifiers have sufficient time to expose malicious behaviours, while avoiding expensive Layer-1 computation in the absence of disputes [31].

Despite these advantages, the OR architecture presents two key challenges that motivate further research [38-41]. First, settlement latency: the mandatory challenge window – necessary for the security deterrence described above – delays the finality of transactions, thereby limiting application scenarios that require real-time settlement [38, 39]. Second, centralisation risks: current implementations rely on a centralised executor to order and execute transactions. This introduces the risk of transaction censorship (where the executor excludes specific users' transactions) [40, 41].

For demand-side smart-grid applications, these limitations are particularly restrictive. Specifically, control and optimisation-based demand-side energy applications often involve compute-intensive optimisation over network constraints. However, under the 'reveal-then-challenge' design, the disputed computational outcomes may be re-executed on Layer-1, which is costly, and often necessitates even longer challenge windows to discourage disputes. As a result, this 'reveal-then-challenge' design with its subsequent long challenge windows directly conflicts with the grid's need for real-time operational responsiveness [37].

Chapter 3 thus proposes a dual-blockchain system with an improved optimistic rollup (IOR) to resolve these identified challenges. The IOR-based dual blockchain design provide the vertical scalability on throughput for the blockchain-based information infrastructure.

2.2.2 Reviews on Consensus Protocols

A consensus protocol consists of a set of pre-specific deterministic operational and communicational rules that enable distributed blockchain nodes to consistently agree on an identical, totally ordered transactions in each log (or block), forming the backbone of blockchain security [43]. Classical Byzantine Fault-Tolerant (BFT) protocols, such as PBFT [44], Tendermint [45], and HotStuff [46], rely on a leader-based architecture. In this model, a designated leader collects transactions, orderly packages them into a block, and broadcasts the block to a validator set containing multiple blockchain nodes. While the leader-based block broadcast design is easy to implement, the leader acts as a significant communication bottleneck [43]. Specifically, as the leader is the sole broadcaster for its generated block, the system's total throughput is strictly bounded by the leader's bandwidth capacity. Furthermore, as the validator set grows, the leader must disseminate the full block payload to an increasing number of nodes; this centralised bottleneck causes latency to spike, particularly in geo-distributed settings where physical propagation delays are naturally higher [43].

To remove this single point of bandwidth bottleneck, research has shifted toward leaderless Byzantine Fault Tolerance (LBFT) protocols. Their core principle is to allow all nodes to broadcast transaction blocks concurrently, utilising the network's aggregate hardware capacity and thus improving throughput horizontally [47-51]. However, their fully decentralised nature introduces two critical side effects that hinder real-world deployment. The first is Redundant Broadcasts (RB) [52, 53]. In conventional LBFT implementations, multiple nodes independently include identical transactions in their blocks. This results in a communication explosion where the same transaction payload is disseminated repeatedly, consuming unnecessary system bandwidth and constraining effective throughput. The second critical challenge in LBFT protocols is Transaction Censorship (TC), which becomes particularly acute under specific adversarial conditions [54]. Existing security analyses typically adopt a 'static corruption model', assuming that a node is either permanently honest or permanently malicious throughout

the protocol's execution. However, this simplification fails to capture the capabilities of mobile adversaries who can corrupt every node over time [55-57].

To Address the dual challenges in LBFT protocols, Chapter 4 introduces the TBG mechanism which simultaneously mitigates redundant broadcast and prevents transaction censorship under dynamic attacks. The TBG-based LBFT protocols provide the horizontal scalability on throughput and robust security against cyber-attacks for the blockchain-based information infrastructure.

2.2.3 Reviews on Cross-Chain Protocols in Multi-Blockchain Systems

The rapid expansion of the blockchain ecosystem has resulted in severe fragmentation, which turns isolated blockchain networks into 'data and value silos' [32, 58]. This necessitates cross-chain protocols capable of transferring assets and state data across different blockchains without compromising these blockchains' security and their operational independence [59]. The literature typically categorises these solutions into three families: (i) Atomic Swaps, which facilitate the direct, peer-to-peer exchange of assets between untrusted parties located in different blockchains; (ii) Relays and Bridges, which establish bidirectional communication channels between two specific blockchains via centralised third parties; and (iii) 'Blockchain-of-Blockchains' (BoB) architectures, which introduce a dedicated connector blockchain to orchestrate complex interactions among heterogeneous and independent blockchains [59].

The foundational approach for atomic swaps is Hash Timelock Contracts (HTLCs), which allow two parties in different blockchains to swap assets atomically by deploying conditional swap programs into smart contracts [60-63]. However, HTLCs face three key limitations. First, HTLCs are inherently vulnerable to multiple attacks, including "sore-loser" behaviours, griefing, and other strategic deviations [59, 64, 65]. While these attacks do not influence the safety property of HTLCs – no assets are stolen by attacks, they cause a severe liveness failure which prevents asset swaps from completing [64, 65]. Second, in complex multi-party settings, HTLCs and their variants

rely on a sequential execution model: transactions must be confirmed one after another in a strict order. This causes latency to accumulate linearly with the number of participants, making real-time settlement impossible [60, 61, 63, 65]. Third, HTLCs necessitate specific smart-contract capabilities to deploy the complex conditional swap program, which restricts their applicability in legacy blockchains (such as Bitcoin), which lack the smart-contract functionality required for complex logic [66-68].

To support richer cross-chain interactions beyond asset swaps, another line of research designs relay-based bridges and BoB architectures [66-70]. However, these schemes face the two key issues that hinder their applicability in real-world setting: First, each blockchain in the relay-based or BoB-based multi-blockchain system (MBS) needs to store extra information of other connected blockchains and the relay (or BoB), which results in the memory scalability issue as the number of connected blockchains increase [71-73]. Second, the state-of-the-art relay-based or BoB-based cross-chain protocols are inefficient, as they need at least multiple phases (at least 4) to confirm a cross-chain transaction, where each phase is equal to a block generation duration of the slowest blockchain in the MBS [73]. The long transaction confirmation latency will hinder their applicability in multiple application scenarios, such as P2P energy trading.

Chapter 5 proposes two cross-chain protocols to resolve the above identified challenges existing in the state-of-the-art cross-chain protocols [74]. The CCTE protocol is designed for two-party asset swaps, which resolves the three key challenges inherent in HTLCs. Second, the CCDI protocol is proposed to resolve two challenges identified above. The two proposed cross-chain protocols provide efficient and secure interoperability for the blockchain-based information infrastructure.

2.3 Review on Distributed Optimisation-based P2P Market Clearing

The proliferation of distributed energy resources has transitioned passive consumers into active prosumers, necessitating P2P energy markets to optimally match bids and offers subject to grid constraints [75-78]. To mitigate the single-point-of-failure risks

inherent in centralised clearing, Distributed Optimisation (DO) algorithms have emerged as a promising solution [77]. By decomposing the global clearing problem into local subproblems, DO enables each participant to independently execute their own assigned subproblems and collectively exchange their resulting updates iteratively.

However, deploying DO in real-time markets (e.g., 5-minute intervals) faces a fundamental trilemma, as existing solutions designed for DO-based energy market clearing cannot simultaneously satisfy three critical requirements [79-81]:

Privacy Preservation: The iterative process mandates that prosumers frequently exchange coupling variables (e.g., traded energy quantities). Research confirms that adversaries can exploit these intermediate variables to launch inference attacks, successfully reconstructing sensitive private data such as load profiles and grid parameters [80, 82]. While cryptographic methods like Homomorphic Encryption enable computation over ciphertexts to prevent this leakage, they incur prohibitive computational overheads, which makes real-time market clearing infeasible [83-86].

Optimal Convergence under Attacks: Systems must guarantee DO's optimal convergence even when Byzantine nodes arbitrarily misbehave to bias each iteration's updated results, e.g., sending incorrect updates or withholding correct updates [87]. Current defences fail due to latency bottlenecks. Conventional blockchain-based solutions are impractical because they necessitate the execution of consensus protocols in every DO iteration to ensure the integrity and correctness of resulting updates, introducing unacceptable delays [88, 89]. Similarly, while Trusted Execution Environments (TEEs) offer efficient hardware-protected computation, their provided trust is established via remote attestation [90]. Performing this attestation step for thousands of iterations creates cumulative latency that exceeds real-time deadlines [91].

Time Efficiency: Therefore, no existing approach can balance the strict timeliness required by energy markets with robust privacy and Byzantine fault tolerance.

Chapter 6 integrates TEEs with the proposed advanced blockchain system to ensure energy data privacy. To tackle the identified issues in TEEs, a D-TASK protocol and a TEAR-DO mechanism are further proposed. By combining the two protocols with the

TEE-based advanced blockchain system, this Chapter is the first to resolve this trilemma, thereby ensuring privacy of prosumers' energy profiles, guaranteeing DO convergence even in the presence of Byzantine nodes and accelerating market clearing speeds by an order of magnitude to ensure real-time P2P market clearing.

2.4 Review on Power Load Forecasting

The transition of power grids from centralised topologies to decentralised networks populated by distributed energy sources has necessitated accurate, privacy-preserving Power Load Forecasting (PLF). While machine learning is essential for capturing load variability, its centralised implementation creates single points of failure and privacy risks [92-96]. Consequently, Federated Learning (FL) has emerged as the paradigm of choice, enabling clients to collaboratively train forecasting models while keeping their sensitive energy datasets local and unexposed [97]. However, applying FL to PLF systems encounters a critical dilemma between security and time efficiency.

First, PLF models require frequent, timely retraining to adapt to intermittent distributed energy sources. Yet, current FL architectures – whether centralised or blockchain-based – suffer from severe communication bottlenecks. In each iteration, a central aggregator (or blockchain leader) must download massive model updates from hundreds of clients and broadcast the global model back, creating an ingress/egress bandwidth bottleneck that throttles model training speed [98].

Second, this centralisation vulnerability of FL extends to security. The central server in the FL framework is the prime target for Byzantine attacks [99]. While integrating blockchain can mitigate server-side risks, it exacerbates communication latency due to the extra consensus overheads introduced by blockchains. More seriously, blockchains fail to defend against client-side poisoning attacks, as they can only defend against Byzantine blockchain nodes [100, 101]. Thus, no existing framework simultaneously ensures communication efficiency, privacy, and robust Byzantine tolerance at both client and blockchain node sides.

To break this dilemma, Chapter 7 proposes CTP-FL which integrates the advanced blockchain system with additional two protocols to support FL-based PLF. The advanced blockchain system provides the ability to defend against attacks from Byzantine blockchain nodes. Supported by the leaderless architecture of the advanced blockchain system, the proposed message encoding protocol resolves the communication overheads originally from the central server or the blockchain leader node. Finally, by integrating a commitment-based hybrid consensus (leaderless && leader) protocol with the advanced blockchain system, CTP-FL achieves the Byzantine tolerance capability at the client side.

CHAPTER 3

Dual-Blockchain with Improved Optimisation Rollup Mechanism

Real-world blockchain-based applications face a critical dilemma: they require frequent, fast transaction handling capability, but existing blockchain solutions are either too slow (due to congestion) or too insecure (due to centralisation). Standard scaling solutions, such as Optimistic Rollups (OR), fail to meet energy trading requirements because they rely on a lengthy "challenge period" (typically 7 days) to detect fraud. To resolve this conflict, this chapter proposes a Dual-Blockchain System integrated with an Improved Optimistic Rollup (IOR) mechanism.

1. Architectural Decoupling: We separate the system into two layers. The Primary Blockchain acts as a lightweight, immutable ledger solely for data storage and final settlement. The Secondary Blockchain selects an executor to execute complex energy settlement tasks off-chain to ensure high throughput.

2. The IOR Mechanism: Unlike traditional rollups that passively wait for fraud challenges (e.g., a week), our IOR mechanism actively validates transactions. It introduces a "Delegated Validator" role: smart meters can privately appoint trusted nodes to immediately verify the computation results from the secondary chain. This effectively eliminates the need for a week-long challenge period while preventing collusion between a pre-selected executor and a validator.

3. Performance Validation: We implemented a prototype using FISCO BCOS and Alibaba Cloud. Simulations demonstrate that compared to single-chain benchmarks, this system significantly reduces gas consumption and transaction execution time. To evaluate its performance under real-world applications, it achieves secure, near-real-time P2P energy transaction settlement suitable for hourly energy markets, overcoming the latency bottleneck of conventional Layer-2 solutions.

This chapter is adapted from: T. Yu, F. Luo, C. Pu, Z. Zhao and G. Ranzi, "Dual-Blockchain-Based P2P Energy Trading System With an Improved Optimistic Rollup Mechanism," IET Smart Grid, vol. 5, no. 4, pp. 246–259, 2022, doi: 10.1049/stg2.12074. The article is available under the Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0): https://creativecommons.org/licenses/by-nc/4.0/. Changes were made to integrate the article into this thesis.

3.1 Introduction

Large-scale deployment of distributed energy resources (DERs, typically wind turbines and solar panels) has been recognised as an important countermeasure to the grand challenges of climate change and ever-increasing energy demand in human society. With integration of DERs, traditional energy consumers, which only take energy from the grid, have been becoming the energy "prosumers (producers-and-consumers)" [102] – meaning that they are capable of both consuming and generating energy. Such a transformation naturally drives the emergence of the peer-to-peer (P2P) energy trading [103], which refers to the paradigm of trading energy directly between end users.

P2P energy trading can be performed in a centralised manner, e.g., set up an energy management system to centrally collect the energy selling/purchase bids from the participants and perform market clearing [4]. Such a centralised design suffers from several limitations such as single point of failure, communication bottleneck, lack of transparency and less scalability. A shared ledger technology called blockchain [104] is recognised as a promising supporting technology that can help to overcome the above limitations in P2P energy trading (depicted as Figure 3-1). Blockchain provides a transparent network environment to enable the participates to reach transactions in a decentralised manner, i.e., without a trusted third-party authority. The transaction data is packaged in groups (known as "blocks") and the blocks are linked sequentially to

form a chain of the data – known as the "blockchain". Newly generated blocks are inserted into a blockchain through a certain consensus mechanism that is mutually undertaken by a group of participants. The blockchain acts as a shared ledger and is synchronised in multiple participants. The data stored in the blockchain is immutable and traceable due to the chain structure and hash pointers used in blockchain (the hash point of a block is generated from the content in the previous block).

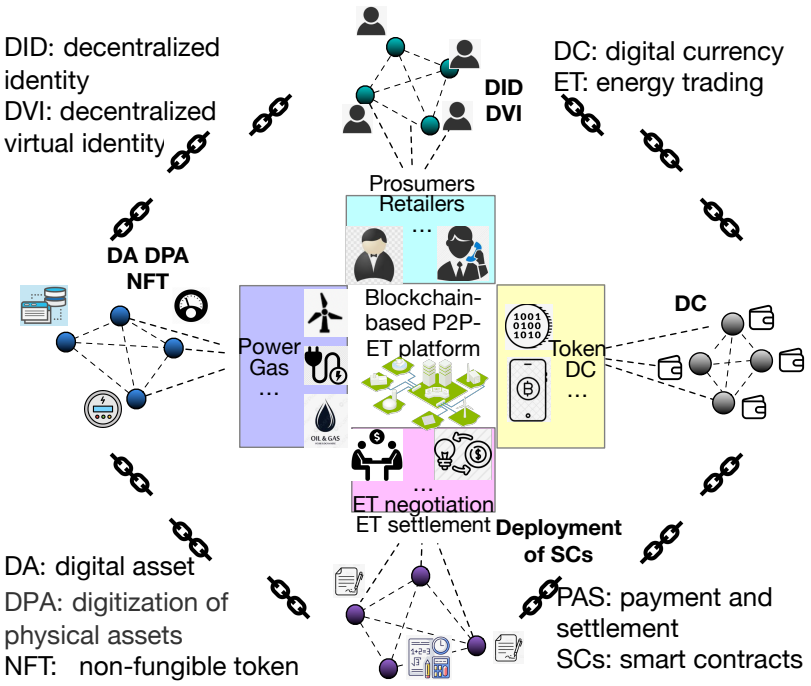

Figure 3-1. Illustration of blockchain-based P2P-energy trading systems.

3.1.1 Challenges of Blockchain-Supported P2P Energy Trading

Blockchain has been playing an important role in P2P energy trading. This is reflected in the extensive academic research (e.g., [104-112]) and several real-world implementations (e.g., [113, 114]) in the last few years. In general, blockchain can support P2P energy trading in the following three aspects: (i) Firstly, the decentralised and distributed nature of blockchain largely reduces the single point of failure risk that exists in centralised energy trading systems (i.e., setting up a central marketplace-like system to collect the energy trading data of the participants and perform transactions); this can enhance the cyber robustness of the system. (ii) As a decentralised ledger

technology, blockchain eliminates the need of a trustable 3rd party for managing the energy trading process. The data and transactions recorded in blockchain are immutable and fully traceable without the involvement of a 3rd authority. This can enhance the system's trustworthiness and also reduce the cost of the utility company on managing the P2P energy trading activities in power distribution networks. (iii) Through cryptocurrency and reputation mechanisms that have been widely implemented in blockchain systems, blockchain can effectively incentivise end customers to invest in DERs and participate in P2P energy trading. This can foster distribution side energy economics and promote the deployment of renewable energy. Despite the above merits, two significant challenges can be identified when applying blockchain into P2P energy trading.

The first challenge is the system's throughput. With the increasing prevalence of DERs, it can be expected that there will be high demand for P2P energy trading in future urban systems. However, the operation of a blockchain usually relies on redundant computing and data storage facilities, which limits its scalability and performance when the number of transactions is large [115]. For instance, the Ethereum system only performs 12~30 transactions per second [116].

The second challenge is the system's scalability. Many P2P energy trading mechanisms involve complex market clearing computations (e.g., [110-112]); simply using a blockchain to execute complex computations and validate all the generated transactions implies all the nodes in the blockchain network must be equipped with powerful hardware – this is unrealistic and it hence limits the number of nodes than can join the blockchain. Some literature [113, 114] adopt private or consortium blockchains to enhance the system's scalability for P2P energy trading. In these schemes, only a limited number of nodes (usually the ones with adequate computing power) are allowed to govern the blockchain and perform complex computations, but this strategy degrades the decentralisation nature of blockchain and therefore affects the system's security.

Rollup mechanisms, which have been implemented in several blockchain systems [123-125], can provide a support to the above issues. With rollup mechanisms, complex

computation tasks are decoupled from the blockchain to be executed off-line, and the computation result is sent back to the blockchain to be stored. To ensure the correctness of the computation result, rollup mechanisms require the set-up of a "*challenge time*" [122], during which any participant can verify the computation result and propose challenges to the incorrect result it has detected. In order to make the incorrect result, which would be produced by malicious participants, can be eventually detected, the challenge time needs to be sufficiently long (usually a week) [122]. Such a long challenge time setting makes the conventional rollup mechanism can hardly suit to the P2P energy trading environment where the transactions need to be settled on time (e.g., on minute or hourly basis).

3.1.2 Contributions of This Chapter

Aiming at addressing the above challenges, this paper proposes a new blockchain system for supporting P2P energy trading. The system is featured by a dual blockchain design, including a primary blockchain and a secondary blockchain. The primary blockchain is responsible for performing the data storage operation that is less compute-intensive, and the compute-intensive tasks in energy trading are executed by specific executors in the secondary blockchain. Compared with the existing P2P energy trading oriented blockchain systems (reviewed in Section 3.2), the proposed system is innovative in the following aspects:

1. The proposed dual-blockchain system is featured by a unique, smart contract-centric design. A set of smart contracts are set up in the primary blockchain for storing different kinds of information generated in P2P energy trading activities. Through migrating compute-intensive energy trading computations to the secondary blockchain, the nodes in the primary blockchain are only responsible for executing the data storage smart contracts, and this can be done in a highly computational efficient manner. As a result, such a design can significantly enhance the system's throughput without compromising the decentralisation and security requirements.

2. As a part of the dual-blockchain system, an Improved Optimistic Rollup (IOR) mechanism is proposed. In the IOR mechanism, for each block generated in the blockchain, an executor is specified to settle the energy trading transactions in the secondary blockchain and uploads the settlement result to the primary blockchain. Then, a validate node is triggered to verify the result. A uniqueness of the mechanism is that the smart meter nodes, which represent the energy traders, can privately specify some nodes they trust to act as validators (called "*delegated validators*") and verify the transaction settlement result. With the assumption that at least one of the delegated validators can be honest and trustworthy to the traders, the incorrect settlement result can be promptly detected. As a result, such a design can implement fast transaction settlement without setting up a long challenge time; in the meantime, it also ensures the reliability and trustworthy of the system.

The rest of this Section is organised as follows. The related work is presented in Section 3.2. The design details of the proposed system are discussed in Section 3.3. The workflow of executing energy trading transactions in the proposed system is presented in Section 3.4. Section 3.5 reports the simulation and Section 3.6 concludes the Chapter.

3.2 Related Work

The system developed in this study is based on a dual-blockchain structure and an IOR mechanism. Hence, in this Section we provide a review for the state-of-the-art research in these mechanisms as well as the representation work on blockchain-based P2P energy trading systems.

3.2.1 Payment Channel Network (PCN)

PCN [117] sets up a separate communication channel aside the blockchain to facilitate a seller and a pair of traders (i.e., a buyer and a seller) to perform trading multiple times. Only after the multiple transactions complete, the information of the transactions (e.g., the balances and digital signatures of the traders) is sent to the smart contract in the

blockchain. If the information is with both traders' signatures, it will be regarded as valid and recorded into the blockchain. Otherwise, if the information is only with the signature of one party, PCN will wait for a challenge time before recording the information into the blockchain. During the challenge time, if the system receives challenges from other nodes (e.g., from the other trader, implying that the trader does not agree with the submitted transaction information), the system will check the time stamps of all the off-chain stored transactions in the submission to determine which transactions should be recorded into the blockchain.

With such a design, a large number of transactions can be executed in the communication channel, and only the final information after a series of transactions needs to be uploaded to the blockchain. This could significantly enhance the system's throughput. Representative real-world implementations of PCN include the Bitcoin-based Lightning network [118] and the Ethereum-based Raiden network [119]. Nevertheless, one major limitation of PCN is that to make sure the transactions with only one-party signature are valid, the challenge time needs to be sufficiently long (typically 1 week); this makes it would take a long time to record the transactions that have been done.

3.2.2 Sidechain

Sidechain [120] uses a dual-blockchain design to reduce the overhead cost of the blockchain and enhance the system's throughput. A representative real-world implementation of the Sidechain mechanism is Plasma [35]. As the name "Sidechain" implies, all the transaction validation and recording work in the Sidechain system are shared by 2 chains: a main chain and a side chain. The main chain is governed by a relatively large number of nodes, and it records the information of the transactions that are performed in it. The side chain is governed by a small number of nodes; it is created by specific smart contracts in the main chain. A node in the main chain can also be a node that governs the side chain as long as it puts a certain number of digital coins (also

known as “tokens”) in its digital account in the main chain. A node can also transfer tokens from its digital wallet in the main chain to the side chain to participate in trading activities in the side chain. The transactions in the side chain are packaged into blocks and is chained, and the head of each block (which is the hash of the data in the block) is sent to smart contract in the main chain. Similar to PCN, Sidechain also sets up a challenge time for each block head from the side chain. If a challenge is received within the challenge time, then the main chain will verify the correctness of the block head (i.e., the hash value) based on the data in the block.

The main merit of such as dual chain-based mechanism is that since the number of nodes in the side chain is much smaller than that in the main chain, the system can have a large throughput in terms of efficiently processing a large number of transactions and generating blocks. However, Sidechain inherently has the same limitation with PCN: the challenge time is usually long, making the transaction cannot be recorded on time. In addition, due to the fact that the number of nodes governing the side chain is small, the Sidechain system is vulnerable to the censorship attack [121]. That is, a part of the nodes in the side chain would collude to maliciously reject packing and executing some transactions.

3.2.3 Rollup-based Mechanisms

In rollup-based blockchain systems [122], all the computational tasks and data are stored in specific smart contracts. Every time when a new block is generated, the computational tasks and data in the smart contract, which are related to validating the new block, are copied and executed off-line; the execution results are then rolled up to the blockchain and are stored there. Rollup-based mechanisms can be generally divided into 2 types: Optimistic Rollups (OR) [123] and Zero Knowledge-based Rollups (ZKR) [124]. The major real-world implementations of OR include Arbitrum [39] and Optimism [125], while the representative real-world ZKR projects are zkSync [126] and StarkWare [127].

OR is based on the fraud-based rollup scheme: similar to PCN and Sidechain, OR also sets up a challenge time for other validators to verify the correctness of transactions. But unlike PCN and Sidechain, all the data and execution codes are not stored off-chain but are stored in specific smart contracts in the blockchain. In this way, if a validator challenges a transaction result, the controversial result can be immediately executed again based on the data in the smart contract. This can thus enhance the system's computational efficiency. Despite this merit, an OR system shows limitations in detecting the collusion between the transaction executor and validator. That is, if a validator deliberately does not report the wrong transaction result during the challenge time, then the wrong result will be recorded into the blockchain. One countermeasure for this is to set up a sufficient long challenge time: Based on a reasonable preliminary that a malicious transaction executor can hardly collude with all the validators in the system, if the challenge time is sufficient long, at least one honest validator would challenge the wrong result – this is why this mechanism uses "optimistic".

Just like the OR scheme, the ZKR scheme also stores the computational tasks and data in smart contracts and execute them off-line. But unlike OR, the execution results in ZKR are not uploaded to the blockchain immediately; instead, the off-line executor uses the zero-knowledge proof [128] to generate a proof for the execution; then, the executor uploads the computation results and the proof to the blockchain. Then, the nodes in the blockchain verify the computation results based on the uploaded proof and results and the raw data that is stored in the blockchain. The above whole process is called the proof-of-validity. With this design, ZKR does not need to set up challenge time to verify the results. The major limitation of ZKR is that zero knowledge proof is usually applied to simple computational tasks; for complex tasks, considerable computing power and time will be consumed to generate the proof, and this limits its applications in many real-world scenarios.

In summary, PCN and Sidechain improve throughput by moving part of the transaction process outside the primary blockchain, but they have data-availability and dispute-resolution limitations because dispute resolution may require data that are not

already available to the primary blockchain. ZKR avoids a challenge window, but proof generation can be computationally intensive for complex settlement tasks. OR makes disputed data available to smart contracts, but its long challenge window and centralised executor can conflict with timely P2P energy settlement. These limitations motivate the IOR design.

3.2.4 Comparison with Permissioned Multi-Channel Architectures

Hyperledger Fabric [230] is a relevant permissioned-blockchain point of comparison because it supports multiple channels among overlapping organisations. In Fabric, a channel is a ledger and communication domain visible to authorised members. It has channel-scoped membership, ledger data and code, together with a configured ordering service achieved by a specific consensus protocol. A channel therefore primarily provides confidentiality and workload separation by partitioning participants, transactions and applications. This can improve privacy and distribute workload when different regional markets or participant groups can be placed in different channels.

The proposed dual-blockchain system addresses a different design question. A channel-based design separates a permissioned network horizontally: different transactions or participant groups are assigned to different channel ledgers. The proposed system separates a single P2P energy settlement workflow by task. The primary blockchain records transaction data and transaction computation results (TCRs) and arbitrates controversial TCRs. The secondary blockchain executes compute-intensive settlement tasks. IOR then checks the returned TCRs through validators and delegated validators. Thus, the channel-based design adopts an horizontally scalable philosophy, where a blockchain is divided into multiple channels, and each channel can be regarded as a small blockchain, responsible for specific application tasks. In contrast, the IOR design adopts a vertically scalable philosophy, where we add a secondary blockchain for the primary blockchain to perform the heavy computational tasks originally undertaken by the primary blockchain.

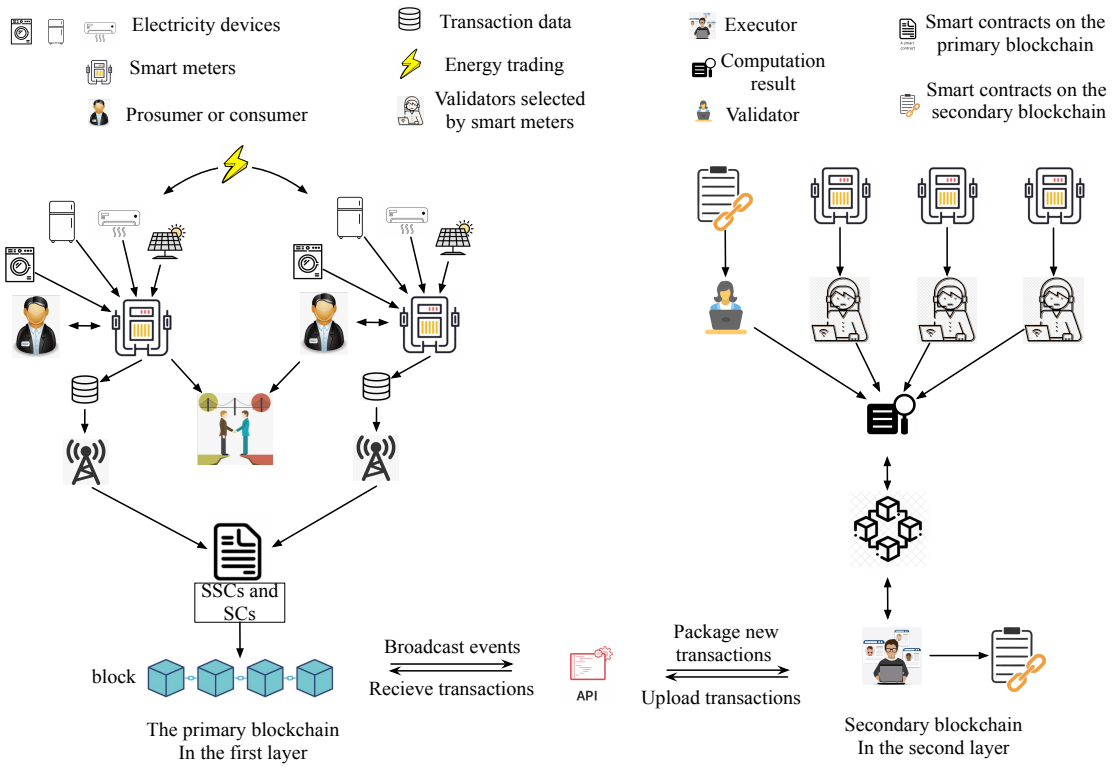

FIGURE 3-2. Architectural design of the proposed dual-blockchain system for P2P energy trading.

Consequently, Fabric channels and the proposed system are complementary. In a practical deployment, Fabric-style channels could be used within an implementation of the primary or secondary blockchain to isolate regional P2P markets, participant groups or privacy domains. IOR would still be needed when compute-intensive settlement results produced outside the primary blockchain must be checked quickly without a long challenge window.

3.3 Dual-blockchain System with IOR Mechanism for P2P Energy Trading

This Section presents the proposed dual-blockchain system with an improved optimistic rollup mechanism for supporting P2P energy trading in low-voltage power distribution networks. The system consists of two blockchains: (i) a primary blockchain for storing the P2P energy transaction data, and (ii) a secondary blockchain for performing

compute-intensive tasks in energy trading. Figure 3-2 illustrates the schematic of the proposed system, and the design details of the system are presented as below.

3.3.1 Classification of Transactions in P2P energy Trading

In the proposed system, energy trading transactions are submitted by smart meter nodes, which act as energy traders and the users of the system. The system divides the transactions generated in P2P energy trading into 2 types: agreement transactions (ATs) and settlement transactions (STs). An AT includes the energy trading information that has been agreed by the energy seller and the buyer. An AT consists of the following tuples:

$$EP||EM||ADDR_b||ADDR_s||DT||DL \tag{3.1}$$

where EP and EM represent the agreed energy trading price ($/kWh) and amount (kWh), respectively. ADDRb and ADDRs represent the buyer and the seller's digital accounts, which are used to transfer and receive the digital coins. DT represents the agreed energy delivery period. DL is the deadline for the buyer to submit its signature to the seller. The system is designed in the way that an AT can only be submitted by the buyer (not the seller) of the energy trading.

A ST contains the information of actual energy delivery from the seller to the buyer. A ST is composed of the following tuples:

$$EP||EM||ADDR_b||ADDR_s||DT||DL||STMP||SIG_b||O_b||\Delta t||\boldsymbol{P}s \tag{3.2}$$

where *STMP* represents the time stamp of the *ST*; SIG_b represents the buyer's signature; O_b is a binary variable attached with buyer's signature, representing the buyer's opinion on the actual energy delivery from the seller: 0-disagreed, 1-agreed. Δt represents the data recording interval of the smart meter (hour). $\mathbf{P}s=[P_{s,1},\ldots,P_{s,DT/\Delta t}]$ is an array, representing the net-power readings of the seller over the energy delivery period. A ST can only be submitted by the seller (not the buyer).

3.3.2 Primary Blockchain

The primary blockchain acts as a decentralised ledger for P2P energy trading. Unlike conventional blockchain-based energy trading solutions (e.g., [105]) in which all the transactions and complex computation tasks deployed in smart contracts are executed on chain, the nodes in the primary blockchain are only required to execute simple tasks, such as storing the energy trading transactions' raw data and computation results, verifying the transaction submitter's identity, and arbitration tasks for the controversial transactions. When a transaction is submitted to the primary blockchain, the nodes in the primary blockchain will store the transaction in their local transaction pools and verify the signature of the transaction sender. Then, the nodes will cache the information of the valid transactions to specific smart contracts which are only used to store raw information of the valid transactions. When the number of the valid transactions in the smart contracts reach a limit, the transactions are packaged into a block. The block is then inserted into the primary blockchain through a consensus mechanism below.

In this study, we use an improved Practical Byzantine Fault Tolerance (PBFT) consensus algorithm [129] to generate blocks. Denote the set of nodes in the primary blockchain as R and the nodes are indexed by integers $\{1, 2, ..., |R|\}$, as proved in reference [44], the primary blockchain is immutable if the following condition is satisfied:

$$|R_h| \geq 2f + 1 \tag{3.3}$$

where f is the maximum number of malicious nodes that are allowed to exist in the system; R_h denotes the number of nodes that behave honestly all the time. Figure 3-3 illustrates the consensus process performed by the nodes in the primary blockchain to generate and validate a block:

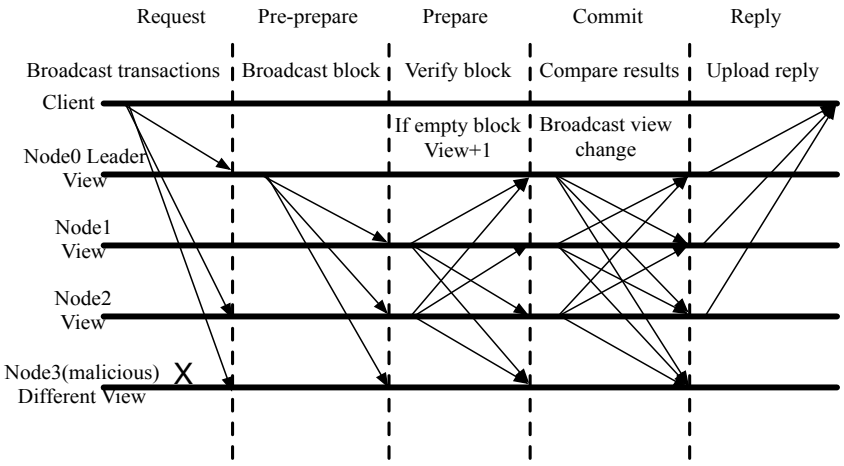

FIGURE 3-3. The process of the improved PBFT consensus.

(1) *Request*. Every time when a new block is generated, a leader is selected from the nodes based on Eq. (3.4). The leader is responsible for receiving the transaction requests send by the energy traders.

$$ID_{lead} = (view + BN) \%|R| \tag{3.4}$$

where *view* is a variable which is initialised to be 1. Every time when the current leader does not complete the new block broadcast task within a required time frame, the value of *view* increments by 1. *BN* is the identification of the newly generated block; ID_{lead} is the index of the leader node (ID_{lead}=1: $|R|$).

(2) *Pre-prepare*. The leader firstly creates a block which contains several transaction requests and sends it to the other nodes. When the other nodes receive the new block broadcast by the leader, they verify the following information: the identification of the leader, the value of *view*, the sequence number of the block, the block's hash, the leader's signature, the hash of transactions, and transaction's execution tasks stored in the block. For an honest node, if the block passes the verification, it will broadcast the signature package generated from the hash of the block to the other nodes in the primary blockchain; otherwise, the honest node will not broadcast anything.

(3) *Prepare*. For each node that has received $2f+1$ signature packages that are identical with the one it has verified and broadcast, it proceeds to perform the step (4);

otherwise, the consensus process terminates for the node.

(4) *Commit*. For each node that has verified the signature packages (specified in Step (3)), it sends a commit message to the other nodes. The commit message indicates the node agrees to insert the block into the blockchain. However, if the block is an empty block, the node will send a "view update" message to the other nodes, indicating that the value of view needs to be updated.

(5) *Reply*. For a node that has received $2f+1$ commit messages from the others, it appends the block into its local copy of the primary blockchain. Otherwise, if it has received $2f+1$ "view update" messages (which indicates the block is empty), the node will ignore the block, increment the value of view by 1, calculate the identity of the new leader according to Eq. (3.4), and wait for starting a new consensus process.

3.3.3 Secondary Blockchain

In P2P energy trading, there could be some compute-intensive tasks in validating transactions, such as verifying the actual energy delivery amount and price. In the proposed system, this part of work will be performed in the secondary blockchain. In the rest of this paper, we call the execution results of the validating transactions as transaction computation results (TCRs). Each node in the secondary blockchain should satisfy the following 2 conditions: (i) it has a certificate authenticated by an authority; and (ii) it has deposited a certain number of coins into the primary blockchain. If a node is detected to misbehaviour in the system, its deposit will be forfeited and the node which discloses such misbehaviour will be rewarded by a certain number of coins.

In the secondary blockchain, a node can be with one of the following 2 roles:

(1) Executors. A group of nodes in the secondary blockchain act as executors. They are responsible for generating and sending TCRs to the primary blockchain. The executors can be dedicated, pre-specified nodes that are equipped with relatively powerful hardware. The executors duplicate all the data that has been stored in the primary blockchain.

(2) Validators. Validators are responsible for verifying TCRs. The validator nodes can be pre-specified. Note that the delegated validators, which will be discussed later, also belongs to the group of validators.

The two blockchains interact with each other as follows: when new transaction data is appended into the smart contracts in primary blockchain, the smart contracts will then broadcast a new data event in the primary blockchain. An executor from the executor group, which in the secondary blockchain, will be notified by the event through a listener it has registered in the primary blockchain. After that, the executor will copy the transaction's data from the smart contracts in the primary blockchain into the smart contracts in the secondary blockchain; it will then execute the smart contracts and upload the TCRs to the corresponding smart contracts in the primary blockchain. Once the TCRs have been recorded in the primary blockchain, the system uses the proposed IOR mechanism to validate the computation result in the primary blockchain and settle the transactions.

3.3.4 Improved Optimistic Rollup Mechanism for Enabling Interaction between Primary and Secondary Blockchains

As discussed in Section 3.2.3, OR [123] has been actively used in blockchain systems to ensure correctness of the off-chain TCRs. The state-of-the-art OR mechanisms rely on a long challenge time and are hence not suitable for P2P energy trading that often regularly performs trading at a short time interval. To tackle this issue, this paper proposes a new OR mechanism (i.e., IOR) to ensure the trust of the system while support fast P2P energy trading transaction settlements.

When the TCRs of a batch of transactions have been recorded in the primary blockchain, IOR is triggered to validate the result. A validator is selected from the validator group to verify the TCRs immediately. If the validator does not have any disputes on the TCRs, it will not send any information to the primary blockchain; otherwise, it will send a "controversial transaction" to a smart contract in the primary

blockchain. The controversial transaction includes the TCRs produced by both the executor and the validator and the original transaction data. Then the primary blockchain will act as the arbitrator and re-execute the controversial transactions to determine which result is correct. With this design, no long challenge time is required to be set, because the TCRs can be verified in the first time by the validators.

There is a possibility that an executor and a validator collude with each other to forge an incorrect TCRs. The IOR takes advantage of the smart meter nodes to avoid such a collusion risk. In P2P energy trading, smart meters record the participants' energy production and consumption. Based on this fact, in IOR if a smart meter detects a part of the TCRs is inconsistent with that computed based on its own recorded readings, it can privately select one or more nodes from the validator group to independently verify the TCRs again. We call such selected nodes as "Delegated Validators". The delegated validators can be selected by smart meter nodes through certain random algorithms, making a third-party entity can hardly predict which nodes will be selected as delegated validators; or they could be nodes the smart meter node which launches the validation trusts. Moreover, a smart meter node can select more than one validator to have them perform validation independently. Therefore, with this design, a malicious executor needs to collude all the validators to make the incorrect TCRs can be successfully recorded into the primary blockchain – this is expensive and unrealistic when the number of the validators is large. Therefore, the proposed IOR can ensure the cyber-security of the system.

Security analysis: We assume all the participants in the system are rationale. An executor could maliciously make profit by transferring the deposited coins by traders to its own digital account. To achieve this, the malicious executor needs to pay bribe fee to each validator in the secondary blockchain with the amount that is larger than the reward the validator can earn from disclosing the malicious behaviour of the executor. Based on the notations defined in Table 3-1, only when the conditions in Eq. (3.5) are satisfied, a malicious executor can make profit:

$$\begin{cases} z_i > \ddot{e} \times y + k_i \times f \\ \sum_{i=0}^{i=m} x_i > n \times z_i \end{cases} \quad (3.5)$$

where $tran_i$ represents the ith transaction of the total transactions the executor has processed. k_i is the number of S Ms who select the same DV_i simultaneously and $0 \leq k_i < m$. In P2P energy trading, the values of both m and n can be very large. Therefore, as long as the value of y is not too small, the bribe cost z the malicious executor needs to pay will be very large, making it difficult for the executor to make profit. This means by properly settings the value of the deposit (y), the security of the IOR-based blockchain system can be ensured.

TABLE 3-1. Nomenclature for IOR

Notation	Meaning
x_i	Profit that the executor can get from $tran_i$
y	Coins deposited by each participant in the primary blockchain
$\ddot{e}$	Bonus coefficient for revealing the misbehaviour of a malicious executor
z_i	Bribe fee paid from a malicious executor to each validator
f	Fee paid by each smart meter to a delegated validator
m	Number of transactions processed by a malicious executor
n	the number of validators in the secondary blockchain

In the rest of this section, we will introduce the execution processes of the two types of transactions (ATs and STs) in the proposed system.

3.3.5 Execution of Energy Trading Transactions in Smart Contracts

Two smart contracts are designed for storing the information related to ATs. For convenient purposes, they are called SC_DS_{AT} and SC_CR_{AT} and they are used for storing the data (Eq. (3.1)) and the computation results of ATs, respectively. In the proposed system, each participant is assigned with a reputation score, which is initialised to be an integer rs. If a participant shows misbehaviours in energy trading, its reputation score will be reduced by a fixed value æ. If the reputation score is below a threshold $ß$, the transactions of the participant will not be considered to be valid. The reputation scores are computed and stored in another smart contract called SC_EXE_{AT},

which contains the execution code of ATs and is processed in both of the primary and secondary blockchains. The details of the execution procedures of SC_EXE_{AT} in both blockchains will be introduced in Section 3.4. An AT is executed with 6 steps:

(1) *Step 1*: Check if the address of the AT sender (i.e., the energy buyer) is equal to $ADDR_b$. If not, proceed to Step 6 to reduce the sender's reputation score; otherwise, proceed to Step 2.

(2) *Step 2*: Check whether the items of *EP* and *EM* in the AT have been signed by the seller. If not, proceed to Step 6; otherwise, proceed to Step 3.

(3) *Step 3*: The nodes in the secondary blockchain access $ADDR_b$ and $ADDR_s$ to verify the traders' balances of digital coins; they also verify the traders' reputation scores. The purpose of the verifications is to determine the validity of the AT. If there is a trader cannot pass one of the two verifications, proceed to Step 6; otherwise, proceed to Step 4.

(4) *Step 4*: Create a new "key-value" pair in SC_EXE_{AT}, where the key is $ADDR_b$ and the value is the data of the AT. Then, proceed to Step 5.

(5) *Step 5*: The buyer and the seller deposit $D_s+EP\times EM$ coins and D_s coins in SC_EXE_{AT}, respectively. The D_s coins will be used as the security deposit for the traders' possible misbehaviours (as introduced later). The execution process terminates.

(6) *Step 6*: Reduce the reputation score of the AT sender by α scores and discard the AT. The execution process terminates.

Two smart contracts are designed for recording the data and the computation results of STs, which are called SC_DS_{ST} and SC_CR_{ST} respectively. A ST is executed in a smart contract with the name of SC_EXE_{ST}. The procedures of executing a ST include the following steps:

(7) *Step 1*: The ST's validity is checked by comparing the values of the data items *EP*, *EM*, $ADDR_s$, *DT*, and *DL* in the ST (see Eq. (3.2)) and the values of these 5 data items stored in the smart contract SC_EXE_{AT} with the key of $ADDR_b$. If the values of these data items in the 2 smart contracts are identical, it can be confirmed the AT and the ST are from the same energy traders. The ST is then considered to be valid and the

execution process proceeds to step 2. If the values are not the same, go to Step 5.

(8) *Step 2*: If the ST does not contain the buyer's signature and the ST's time stamp has surpassed *DL*, the buyer will be punished while the ST will still be executed. $F_p \times D_s$ coins in the buyer's deposit in SC_EXE_{AT} will be transferred to $ADDR_s$ as compensation, where $0<F_p<1$ is a penalty factor, and the rest $(1-F_p)\times D_s$ coins in the buyer's deposit will be "burned" by being transferred from SC_EXE_{AT} to a "black hole" address $ADDR_{BH}$ for which no participants in the system own the private key. The execution process then terminates. Otherwise, if the ST contains the buyer's signature or the ST's time stamp has not surpassed DL, go to Step 3.

(9) *Step 3*: (i) If $O_b=1$, D_s coins deposited by the two traders of the ST will be unlocked in SC_EXE_{AT} and be sent back to the addresses of $ADDR_b$ and $ADDR_s$, and the execution process terminates. (ii) If $O_b=0$, proceed to Step 4.

(10) *Step 4*: A dispute arbitration process is performed based on the seller's smart meter data: If $\sum_{i=1}^{i=\frac{DT}{\Delta t}} \mathbf{P}s \times \Delta t < EM$, the buyer will get additional compensation fee of $F_p \times D_s$ coins from the pledged tokens of the seller, and the remaining $(1-F_p)\times D_s$ coins will be sent to $ADDR_{BH}$. Otherwise, the seller will get $F_p \times D_s$ coins as compensation and the remaining $(1-F_p)\times D_s$ coins from the buyer's deposit will be sent to $ADDR_{BH}$. The execution process terminates.

(11) *Step5:* Reduce the reputation score of the ST sender by α scores and discard the ST. The execution process terminates.

Note that the "burning" mechanism in Step 2 of the deposit is set up for resisting the Distributed Denial of Service (DDoS) attack, in which a malicious node could create two identities to act as a buyer and a seller at the same time and submit tremendous invalid transactions to break down the primary blockchain. With the "burning" mechanism, $n\times(1-F_p)\times D_s$ coins will be incurred as the cost of launching a DDoS attack with n invalid transactions; this will make a rational attacker to give up launching DDoS attacks.

Additionally, there are another 2 smart contracts executed by the validators to

verify the results recorded in SC_CR_{AT} and SC_CR_{ST}, respectively. For convenience purpose, we call them as verify smart contracts (SC_VERIFY).

3.4 Overall Operation Workflow of the Proposed System

Based on the discussion in the previous sections, this section summarises the operation workflow of the proposed system (depicted in Figure 3-4. Operation workflow of the proposed system.). There are 5 major steps in the system's operation:

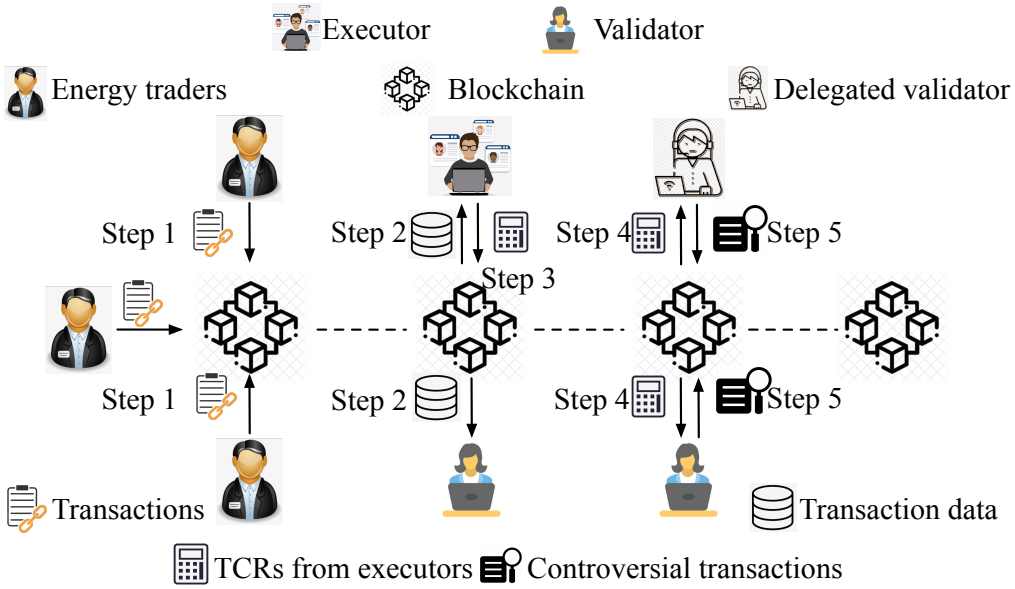

FIGURE 3-4. Operation workflow of the proposed system.

(1) *Step 1*: When multiple transactions are sent to the system, the primary blockchain executes the smart contract SC_DS_{AT} (for ATs) or SC_DS_{ST} (for STs) to record the data in the transactions into the primary blockchain. New blocks are generated, where each block stores the data of a certain number of transactions.

(2) *Step 2*: For each newly generated block, an executor and a validator retrieve the data in the block and execute the smart contract SC_EXE_{AT} (for ATs) or SC_EXE_{ST} (for STs) locally.

(3) *Step 3*: The executor packages its TCRs as a transaction, which subsequentially invokes the smart contract SC_CR_{AT} or SC_CR_{ST} in the primary blockchain. Then, a new block in the primary blockchain is generated to record the TCRs of transaction sent

by the executor in the corresponding smart contracts.

(4) *Step 4*: The validator retrieves the TCRs stored in the block generated in Step 3 and compare it with that generated by itself through executing the smart contract SC_VERIFY locally.

(5) *Step 5*: if the computation results produced by the executor and the validator are inconsistent, the validator sends a *controversial transaction* to invoke the smart contract SC_EXE_{AT} or SC_EXE_{ST} in the primary blockchain. The corresponding smart contract will be executed to determine which TCRs are valid. This step can also be performed by delegated validators.

3.5 Experiments

Simulation is conducted to validate the proposed system. The implementation of the proposed blockchain system and the simulation result of the blockchain-based P2P energy trading are reported and discussed in the following sections.

3.5.1 Implementation of the Blockchain System

We implement the proposed system based on an enterprise blockchain project FISCO BCOS [129]. FISCO BCOS is an open-source blockchain platform that supports the deployment of customised smart contracts subjected to different application scenarios. It provides an interactive, scripting environment for the user to build blockchain, develop smart contracts and monitor the system's performance. We use the Solidity scripting language to program the smart contracts presented in the previous sections in FISCO BCOS, which are executed in the Ethereum virtual machine. We implement the primary and secondary blockchains using the fundamental components (e.g., the PBFT consensus, the P2P communication protocol and the RocksDB storage engine) provided by the platform. The maximum number of P2P energy trading transactions that can be recorded in a block in the primary blockchain is set to be 300.

We deploy the blockchain system built in FISCO BCOS in a virtual server created

in Alibaba Cloud [130]. The server is configured with the Ubuntu operating system, 120 Gigabyte storage, 5Mbps bandwidth, 16 cores, and 32-Gigabyte memory. Alibaba Cloud provides interactive interfaces to manage the computing resources in the cloud in an on-demand manner.

3.5.2 Evaluation Metrics

We measure the performance of the system in terms of the following metrics:

(1) *Execution time* of a number of concurrent transactions, which is the time duration between the time when a batch of transactions are submitted concurrently and the time when all of them are recorded into the blockchain.

(2) *Throughput*, which means the rate at which the valid transactions are recorded into the blockchain, i.e., the number of recorded valid transactions per second.

(3) *Gas consumption unit*. The gas consumption reflects the computational intensity of a task in smart contracts. Since the gas consumption amount is usually a large number, for convenience purposes, we define the amount of 124,000 gas as 1 unit – in the implemented prototype system, this is the amount of gas consumed by executing the smart contract SC_DS_{AT}. The gas consumption units of all the smart contracts in the system are shown in Table 3-2 and Table 3-3.

TABLE 3-2. Overview of the smart contracts in the proposed system (i)

Type	SC_DSAT	SC_CRAT	SC_EXEAT	SC_DSST
Location	Primary blockchain	Primary blockchain	Both blockchains	Primary blockchain
Function	Record data of ATs	Record TCRs of ATs	Execute ATs	Record data of STs
Gas consumption	1 unit	0.95 units	6.9 units	1 unit

TABLE 3-3. Overview of the smart contracts in the proposed system (ii)

Type	SC_CRST	SC_EXEST	SC_VERIFY
Location	Primary blockchain	Both blockchains	Both blockchain
Function	Record TCRs of STs	Execute STs	Verify the TCRs from the executor
Gas consumption	0.95 units	11.2 units	2.1 units

3.5.3 Evaluation Result and Discussion

We run the implemented prototype system in the Alibaba Cloud to execute P2P energy transactions. For comparison purposes, we implement a benchmark system in FISCO BCOS and Alibaba Cloud as well, which is a conventional, single-blockchain system. In the benchmark system, the smart contracts for executing ATs and STs, i.e., SC_EXE_{AT} and SC_EXE_{ST} are deployed and the fundamental components such as the consensus protocol, are the same as the primary blockchain in the proposed system.

We compare the performance of the proposed and benchmark systems in terms of the aforementioned metrics. We simulate 1,000 concurrent P2P energy trading transactions, and Figure 3-5 shows the comparison of the total execution time in executing the 1,000 transactions in the primary blockchain under the proposed and the benchmark systems. As shown in Figure 3-5, we evaluate these metrics in both systems by setting the number of nodes in the primary blockchain (or in the sole blockchain in the benchmark system) to be 10, 20, and 30 configured with 0.5 CPU cores, respectively. Figure 3-5 shows that in the proposed system, it only takes about 2.7 seconds, 3.4 seconds, and 4.3 seconds for executing the 1,000 concurrent transactions in the primary blockchain with 3 different sizes, which are significantly lower than those in the benchmark system. This is because in the proposed system, the primary blockchain only needs to execute SC_DS_{AT} and SC_DS_{ST} for recording the data from ATs and STs, and

each task only consumes 1 unit of gas; in the benchmark system, the blockchain needs to execute SC_EXE_{AT} and SC_EXE_{ST} for executing ATs and STs, and these two tasks consume 6.9 and 11.2 units, respectively. The results clearly show that by mitigating compute-intensive tasks into the secondary blockchain, the computational efficiency of the primary blockchain can be largely enhanced.

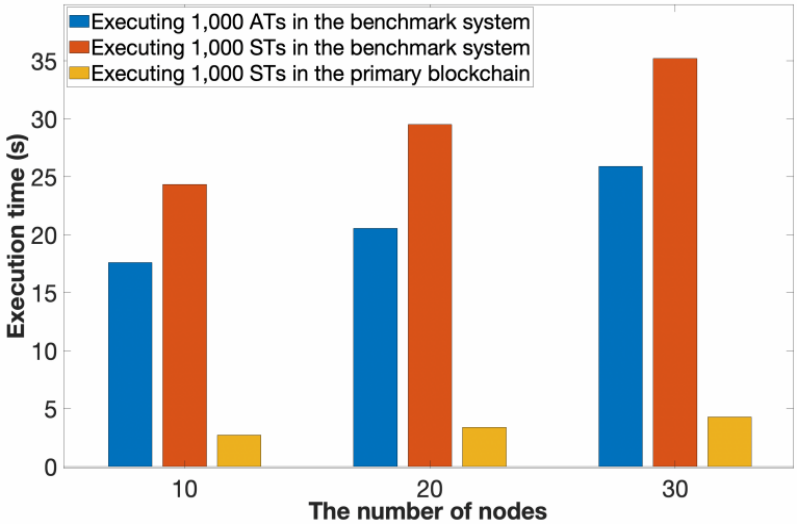

FIGURE 3-5. Comparison of the primary blockchain and the benchmark system on the execution time.

The throughput of the system significantly depends on the number of the nodes in the primary blockchain and the computing power of the nodes. Figure 3-6 shows the measured throughput of the system implemented in FISCO BCOS and Alibaba Cloud with 5-node and 10-node configurations. It can be clearly seen that with the increase of the number of CPU cores in the cloud, the throughput of the system increases in an approximately linear manner. Also, when there are more nodes in the primary blockchain, the computational cost of the PBFT mechanism will increase; as a result, the system throughput will decrease. For example, when the number of CPU cores is 1.5, the system throughput in the 10-node is about 706tps, while that in the 5-node case is about 804tps. In real-world applications, the number of nodes in a blockchain is usually larger than 10, and this will further decrease the system throughput. From the authors' perspective, when practically applying the proposed system, the throughput

can be expected to be larger than 1,000tps through properly configuring the computing power and the number of nodes in the primary blockchain.

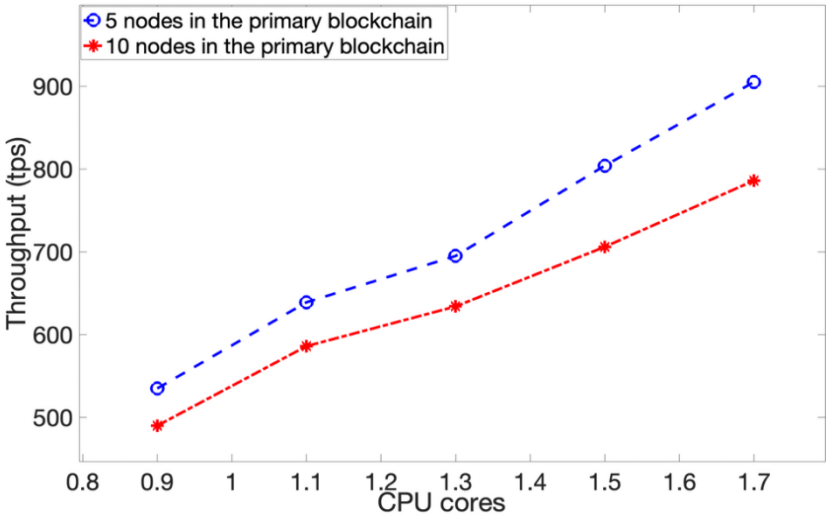

FIGURE 3-6. Throughput test with the change of CPU cores.

The total execution time of the 1,000 STs in the proposed system is roughly 17.5 seconds, in which about 15 seconds are spent on executing and verifying the STs and around 2.5 seconds are spent on generating blocks and internal communication. Obviously, this computation performance can satisfy the requirement of P2P energy trading where the trading interval is usually on hourly or minute basis.

We further compare the gas consumption of the benchmark and the proposed systems. Figure 3-7 shows the units of gas consumed by the 2 systems for processing 1,000 ATs and 1,000 STs under 10-, 20-, and 30-node settings. It can be seen that the amount of gas consumed by the proposed system is significantly less than the gas consumed by the benchmark system. This is because that in the proposed system, the compute-intensive smart contracts (i.e., SC_EXE_{AT} and SC_EXE_{ST}) are only executed by an executor and several validators in the secondary blockchain; in contrast, they need to be executed by all the nodes in the benchmark system. As a result, such as gas consumption difference is more obvious with the increase of the nodes in the blockchain, as that has been reflected in Figure 3-7.

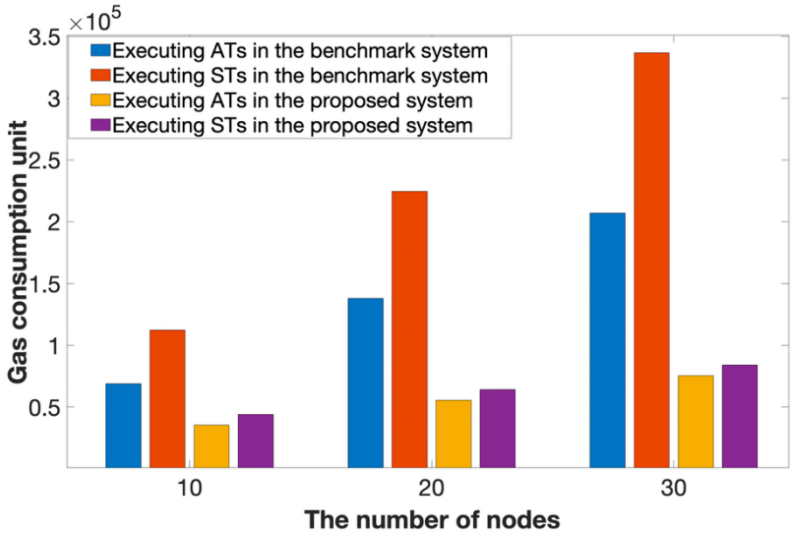

Figure 3-7. Gas consumption unit comparison in both systems.

We further compare the proposed IOR mechanism and the existing OR mechanism (from the Arbitrum system [39]) in terms of gas consumption and transaction settlement time under different number of transactions (this corresponds to STs in the proposed system). In this comparison, the numbers of nodes in the primary blockchain in both systems (i.e., the proposed blockchain system and Arbitrum) are fixed at 10. We simulate 2 situations: (1) without dispute for the transactions, meaning that there are no different opinions from the participants for the transactions; and (2) with dispute, meaning that the system receives disputes for some of the TCRs. In the proposed system, for each disputed ST, 1 delegated validator is selected to perform validation for the disputed ST; in Arbitrum, for each disputed transaction, 1 validator is selected for the same purpose. The comparison result is reported in Table 3-4, Table 3-5, Table 3-6.

The result reveals that the conventional OR is slightly more efficient in terms of gas consumption. For example, in the situation without dispute, when there are 4,000 transactions, the proposed system consumes 178,900 units of gas to process them, while that under the conventional OR is 134,100 units. This additional gas consumption is due to the transaction validation work in the proposed system: as we have presented in the previous sections, in the proposed system, a ST is always validated by the validator(s) in the secondary blockchain and are settled once it passes the validation. In

contrast, in the conventional OR, a transaction will not be immediately settled; instead, the system will wait a certain period (e.g., a week) for any possible disputes. In the meantime, when there are disputes proposed by other participants for a ST, the "delegated validator" mechanism (presented in Section 3.3) in the proposed system is triggered immediately to provide extra validation for the disputed ST; the disputed ST then can be settled after it is validated. This is reflected in the 4th row of Table 3-4, Table 3-5, Table 3-6, which shows that no matter there are disputes or not, the STs can be settled in a short time range. In summary, the comparison result shows that although the proposed IOR mechanism consumes more gas than the conventional OR, it can facilitate the transactions to be promptly settled without being delayed for a long challenge time. This indicates the proposed mechanism is more applicable to the P2P energy trading environment where each participant usually frequently forms energy trading transactions with other participants.

TABLE 3-4. Comparison between the IOR and the conventional OR under 1,000 STs

		1,000 STs		
		Without disputes	With disputes for 2% of STs	With disputes for 5% of STs
IOR	Gas consumption (units)	45,600	47,840	51,200
	Total settlement time (seconds)	17.5	17.9	18.7
Conventional OR	Gas consumption (units)	34,400	36,640	40,000
	Total settlement time (seconds)	A week	N/A	N/A

TABLE 3-5. Comparison between the IOR and the conventional OR under 2,000 STs

		2,000 STs		
		Without disputes	With disputes for 2% of STs	With disputes for 5% of STs
IOR	Gas consumption (units)	90,200	94,680	101,400
	Total settlement time (seconds)	30.5	31.4	33.1
Conventional OR	Gas consumption (units)	67,800	72,280	79,000
	Total settlement time (seconds)	A week	N/A	N/A

TABLE 3-6. Comparison between the IOR and the conventional OR under 4,000 STs

		4,000 STs		
		Without dispute	With disputes for 2% of STs	With disputes for 5% of STs
IOR	Gas consumption (units)	178,900	187,860	201,300
	Total settlement time (seconds)	68.3	69.9	73.6
Conventional OR	Gas consumption (units)	134,100	143,060	156,500
	Total settlement time (seconds)	A week	N/A	N/A

3.5.4 Demonstration of P2P Energy Trading in The Proposed System

As a case for demonstration, we simulate 15 participants of P2P energy trading, which include 12 energy prosumers that are equipped with photovoltaic solar panels and/or wind turbines and 3 pure energy consumers. The power load data of the 15 participants are modified from the Australian “Smart Grid, Smart City” dataset [131]. We use the

solar radiation and wind speed data that have been reported in [132] to simulate the wind and solar power output following models:

$$P_t^{pv} = I_t \times A \times \eta, \tag{3.6}$$

$$A^{bld} = \frac{\pi \times D^2}{4}, \tag{3.7}$$

$$P_t^{wind} = \begin{cases} 0, & \text{if } v_t < v^{in} \\ \frac{1}{2} \times \rho \times A^{bld} \times v_t^3, & \text{if } v^{in} \le v_t \le v^{rate} \\ \frac{1}{2} \times \rho \times A^{bld} \times (v^{rate})^3, & \text{if } v^{rate} < v_t \le v^{out} \\ 0, & \text{if } v^{out} < v_t \end{cases}, \tag{3.8}$$

where P_t^{pv} is the power output of the solar panel (kW); I_t represents the solar radiation density at the *t*th time slot (kW/m^2); η denotes the energy conversion factor of the solar panel (%); A is the surface area of the solar panel (m^2); P_t^{wind} denotes the wind power output at the *t*-th time interval (kW); ρ is the air density and is set to be 1.3kg/m^3; v_t, v^{rate} and v^{out} are the cut-in, rated and cut-out wind speed of the wind turbine, respectively (m/s); A^{bld} is the blade's swept area (m^2) and D is the blade's diameter (m). v_t denotes the wind speed at the *t*th time interval (m/s).

We simulate 194 energy trading transactions among 15 participants. Figure 3-8 shows the distribution of the transactions in a day. A total of 35.98 kWh energy is traded in the 194 transactions; as a result, a total revenue of 7.6$ is created for the 12 prosumers over 1-day operation. A total of 4.2 seconds is spent on processing the transactions in the proposed system. Figure 3-9 shows the profiles of the power load, renewable power and P2P energy trading amounts of a typical prosumer.

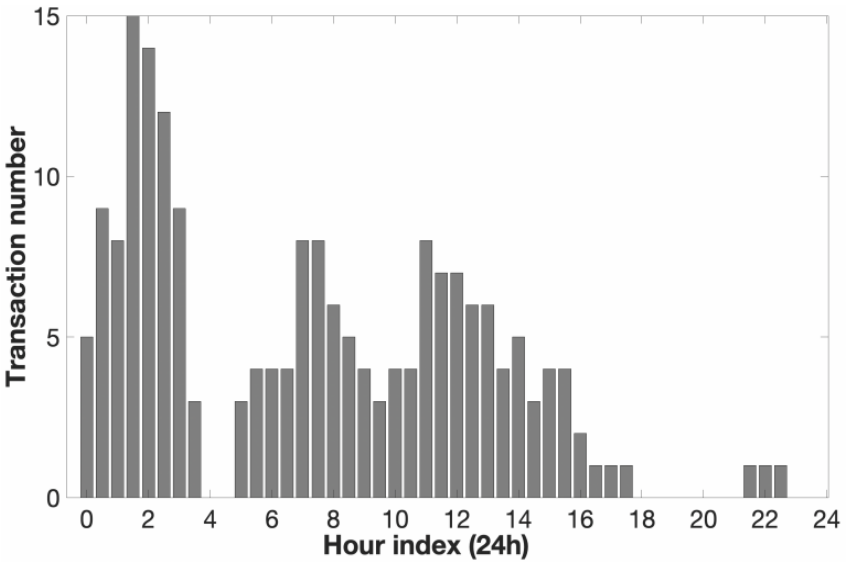

FIGURE 3-8. Distribution of P2P-ET transactions in a day.

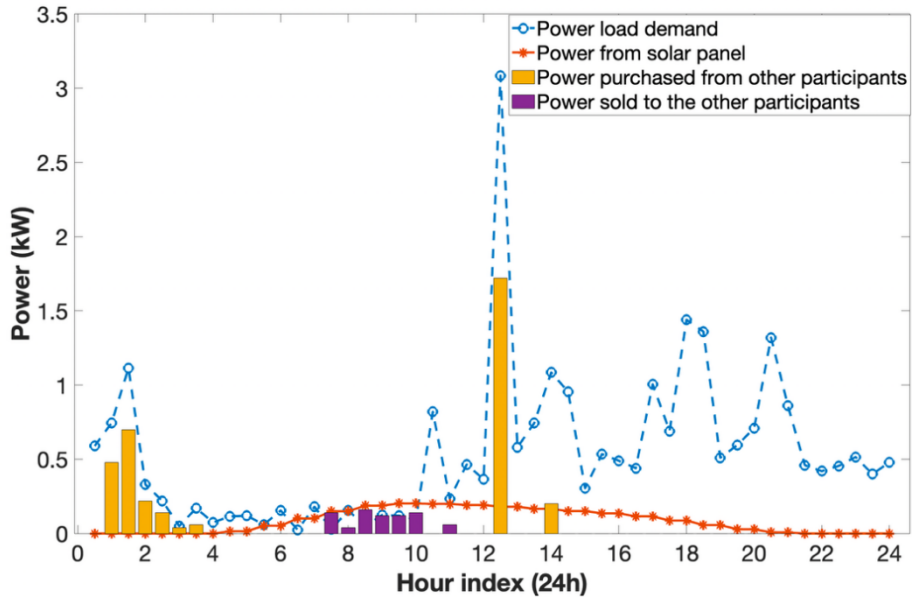

FIGURE 3-9. Profiles of power load demand, renewable power generation, and P2P energy trading amount of a typical prosumer.

3.5.5 Case Study of Reputation Mechanism

We design a simple and representative case to demonstrate how the reputation mechanism in the proposed system works. For the above 15-participant system, we set the participant #14 to be malicious (e.g., it purposely does not output the contracted energy to the buyer). For the reputation mechanism presented in Section 3.3.5, we set the parameters rs, α and β to be 30, 10 and 10, respectively and set D_s and F_p to be 1

and 0.5, respectively (the meaning of these parameters can be found in Section 3.3).

We assume the participant #14 misbehaviours twice, at the time slots of 8 am-8.30 am and 1 pm-1.30 pm, respectively. Therefore, after 1.30 pm, the reputation score of participant #14 is less than 10, making it unable to participate in energy trading anymore. Furthermore, based on the deposit mechanism for SC_EXEAT and the punishment mechanism for SC_EXEST (see Section 3.3.5), the participant is punished twice correspondingly and it loses $2\times D_s=2$ coins. The 2 participants who trade energy with the participant #14 at the 2 time slots then receive compensation of $F_p\times D_s=0.5\times 1=0.5$ coins that are from the deposit of the participant #14 in a smart contract. This case shows the proposed system can effectively benefit honest traders and punish malicious traders, and this can hence enhance the trustworthiness of the system.

3.6 Chapter Summary

This chapter studies blockchain-supported P2P energy trading. It proposes a dual-blockchain system integrated with IOR to separate data recording and arbitration from compute-intensive settlement execution. The primary blockchain stores valid transaction data and TCRs and arbitrates controversial TCRs, while the secondary blockchain executes settlement computation. The experiments in Section 3.5 show lower execution time and gas consumption than the single-chain benchmark under the evaluated settings. The case study further illustrates how the reputation mechanism can penalise trading misbehaviour.

The economic security analysis in Section 3.3.4 assumes rational participants. Under this assumption, properly setting y according to Eq. (3.5) makes executor-validator collusion economically unattractive. For irrational participants that accept economic loss to disrupt the P2P energy trading system, Eq. (3.5) does not provide a sufficient deterrent. IOR can help expose inconsistent transaction computational results. However, IOR does not by itself directly address the attacks launched by irrational

executors. For example, the executors may incentivise validators with higher bribes, even if it results in a negative payoff. Future work should therefore design mechanisms that can provide system security even under irrational participants, such as the Byzantine tolerant mechanisms proposed in Chapter 4, Chapter 6 and Chapter 7.

Although the direct application in this chapter is P2P energy trading, the design may support other transaction-heavy smart-grid applications after adaptation. Such adaptation would need to redefine the transaction format, smart contracts, validator-selection process and dispute conditions for the target application. These extensions are not evaluated in this chapter and remain future work.

CHAPTER 4

Leaderless Byzantine Fault Tolerance Protocols

Leaderless Byzantine fault tolerant (LBFT) protocols enable all replicas to propose batches in parallel, aiming to fully harness all replicas' hardware resources. However, they face two critical drawbacks: redundant broadcasts (RB), which exhausts system bandwidth, and transaction censorship (TC), which undermines system security. Moreover, the issues of RB and TC will be exacerbated under the mobile adversary who can continuously corrupt replicas. To evaluate the mobile adversary in LBFT settings, we introduce a recovery-based corruption model. Our analysis reveals that under the corruption model and a restricted asynchronous network model, the TC rate reaches 76%. To tackle the issues of RB and TC under these models, we propose π^{τ}, a protocol that mitigates RB and TC by eliminating invalid transactions and mapping a valid transaction to solely τ replicas; and develop TBG, a protocol that mitigates RB under adversarial attacks. In simulation, TBG-based LBFT protocols achieve nearly zero TC rate under the proposed recovery-based corruption model, the restricted asynchronous network condition and corresponding attacks. Simulation results show that compared to a hash space method-based LBFT protocol, our TBG-based LBFT protocol improves measured admitted-transaction throughput and simulated latency by up to 14.95× and 12.89× respectively and reduces per-replica memory overhead by up to 33×. Overall, this work takes a first step toward studying LBFT under the recovery-based corruption model, and it motivates scoped protocol-agnostic batch-generation mechanisms as a promising research direction for securing LBFT protocols.

This chapter is based on the author accepted manuscript of: T. Yu, F. Luo and B. Yu, "TBG: A Batch Generation Mechanism for Leaderless Byzantine Fault Tolerance Protocols," accepted for publication in IEEE Transactions on Information Forensics and Security, 2026. This thesis includes the accepted manuscript version, not the final published version.

4.1 Introduction

Leaderless Byzantine Fault Tolerant (LBFT) protocols have gained renewed attention as the core framework underlying decentralised ledgers, e.g., blockchain [133]. These protocols realise the atomic broadcast abstraction [134] – all *correct replicas* that follow protocol rules deliver an identical, totally ordered sequence of transactions despite the presence of *Byzantine replicas* that may arbitrarily deviate from protocol rules. LBFT protocols have two merits compared to other types of BFT protocols (e.g., leader-based BFT protocols [135-140]). First, they allow all replicas to propose their own batches of transactions (or sub-blocks) in parallel within each specified period, known as an epoch. By harnessing the combined hardware capacity of all replicas, LBFT protocols boost overall resource utilisation. Consequently, as the replica count increases, LBFT protocols may process a greater number of transactions per epoch. Second, LBFT protocols ensure that no individual replica can halt its underlying system, thereby negating the need for leader recovery mechanisms [136-138].

Despite their theoretical robustness, LBFT protocols face two key challenges. First, replicas independently broadcast batches of transactions, causing the same transaction to be broadcast multiple times – a phenomenon termed as redundant broadcasts (RB) [141-148]. RB wastes precious bandwidth of replicas, which in turns degrades protocol efficiency. As demonstrated in [146], adversaries can exploit RB to degrade system throughput by a factor of N, where N is the number of replicas. Second, transaction censorship (TC) can arise in LBFT protocols [141, 149]: if a transaction is received only by Byzantine replicas, the transaction can never be broadcast, which undermines system security. We now review the state-of-the-art solutions designed for mitigating the two challenges.

4.1.1 Countermeasures on Redundant Broadcasts

There are two main approaches to mitigate RB. One approach, the random selection

model [141], mitigates RB by having a replica randomly select t transactions from its memory containing T transactions. However, its effectiveness diminishes significantly as t approaches T. Another approach, as in [145-147], maps a transaction tx to a hash space by computing the hash of tx's sender identifier (SID) concatenated with tx's timestamp. This hash space-based method (HSM) restricts a replica to processing transactions in only one hash space per epoch, thereby resolving RB. There are also some approaches that mitigate RB by altering the design of LBFT protocols [142-144, 148]. Among them, only HSM can – without any redesign of LBFT protocols – guarantee that each transaction is broadcast exactly once. However, HSM has three key shortcomings:

(i) In LBFT protocols, timestamps of transactions must not be fully deterministic - otherwise the resulting hash space of a transaction becomes predictable, enabling adversaries to delay or censor the transaction [146]. Conversely, non-deterministic timestamps of transactions introduce a new attack surface by allowing malicious senders to manipulate their timestamp values. Specifically, as shown in Figure 4-1(a), a malicious sender can generate M versions of a transaction tx – where M equals the number of hash spaces – each bearing a unique timestamp. By properly selecting each timestamp, the M different timestamps cause every version of tx mapped into a distinct hash space. As a result, every replica will broadcast tx under this context.

(ii) Under an asynchronous network model, HSM fails because a scheduler introduced by the model can impose arbitrary finite delays on messages between any pair of system participants [150]. As shown in Figure 4-1(b), suppose that responsibility for hash space m is assigned to replica 1 in the epoch indexed by e and to replica 2 in the epoch indexed by $e+1$. 1) The scheduler forces replica 1 to remain stuck in epoch $e-1$ while pushing replica 2 to advance to epoch $e+1$. Then, replica 2 broadcasts the transactions in hash space m at the start of epoch $e+1$. 2) The scheduler withholds replica 2's broadcasts and push replica 1 proceed to epoch e. As replica 1 cannot receive replica 2's broadcasts, it broadcasts these transactions in hash space m again.

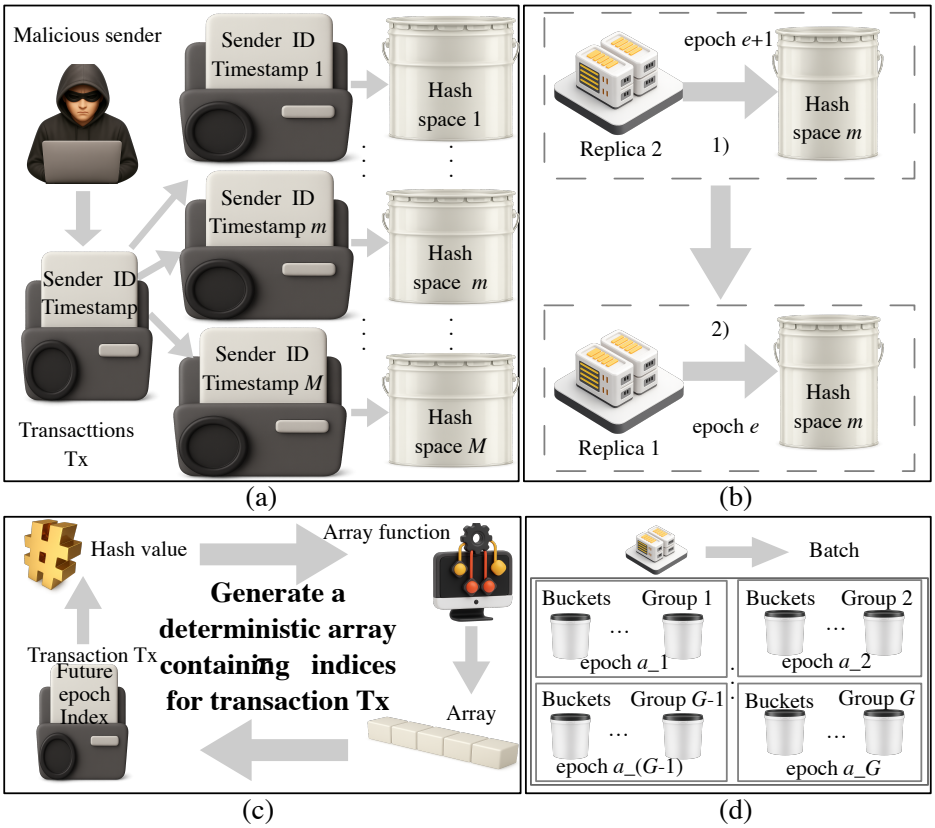

FIGURE 4-1. Illustration of the two attack scenarios and our corresponding defences. (a) Attack model specified in shortcoming (i). (b) Attack model specified in shortcoming (ii). (c) π^{τ}-based method designed for resolving shortcoming (i). (d) TBG-based method designed for resolving shortcoming (ii).

(iii) Since each replica is restricted to broadcasting a limited-size batch (e.g., 2,000 transactions from an identical hash space) under HSM's design, when an onslaught of transactions is naturally or deliberately sent to the LBFT network in each epoch, transactions may accumulate in a hash space and cannot be broadcast promptly.

Although HSM is effective for resolving RB, no prior work considers the attacks on HSM as in Shortcomings (i)-(iii). Accordingly, the first research question is *whether an approach exists that can overcome HSM's three shortcomings*?

4.1.2 Countermeasures on Transaction Censorship

Current countermeasures against TC of LBFT protocols are predicated on two corruption models [151-167]: *Static* (the adversary corrupts f replicas before

execution [151]) and *Adaptive* (the adversary dynamically corrupts up to f replicas during execution [158]). Guided by this f-threshold assumption, protocols like Direct Acyclic Graph (DAG)-based LBFT protocols [159-167] (e.g., Bullshark [152]) or Asynchronous Common Set (ACS)-based LBFT protocols (e.g., Dumbo [142]), systems enforce a "safety in numbers" strategy: a client must send its transaction to at least $f + 1$ replicas. Since the adversary cannot corrupt more than f different nodes under the assumption, at least one recipient remains correct and can broadcast the transaction. With the "safety in numbers" strategy, protocols develop other properties, e.g., data availability [154] and strong validity [142] properties, to ensure the eventual delivery of correct replicas' batches.

However, unlike the static or adaptive models which defines an adversary who have finite budget to corrupt at most f replicas, in real-world setting, there exists a "mobile adversary" who can corrupt every replica over time: by allowing compromised replicas to "recover" (reset to a correct state), the adversary frees up its attack resources to corrupt new targeted replicas continuously [168-173]. In an asynchronous network, this creates a critical vulnerability which we term the "Corruption-Chasing Attack" (CCA):

- **The Delay:** The asynchronous scheduler can delay the specific correct replicas holding a client's transaction (due to network congestion or malicious scheduling).
- **The Chase:** During this delay, a mobile adversary can identify the replicas holding the transaction and sequentially corrupt these identified replicas.
- **The Erasure:** Once an identified correct replica is corrupted, the adversary deletes the transaction from the replica's memory before the transaction is broadcast.

More seriously, because standard asynchronous LBFT protocols typically limit the batch size of each replica per epoch to prevent from Distributed Denial of Service (DDoS) attacks [141], even if a transaction is received by a correct replica, it may not be able to broadcast it immediately when the number of transactions in its memory is larger than the batch size. This further exacerbates the probability of TC under high network congestion periods where each correct replica's memory pool is full of transactions ready to be broadcast.

To our knowledge, no research has investigated LBFT protocols under recovery-based model. Thus, the second research question is that *under a well-defined mobile-adversary corruption model and attacks crafted under the restricted asynchronous network model, is TC rate non-negligible (theoretical gap), and if so, how can we significantly reduce TC rate?*

4.1.3 Contributions

This work addresses the above two key research questions: the challenges posed by RB and the theoretical gap in TC. First, we propose a protocol π^{τ} to solve the shortcomings (i) and (iii). To address shortcomings (i), π^{τ} leverages a *deterministic yet presently unpredictable value* to replace the above non-deterministic timestamps design. As shown in Figure 4-1(c), for a transaction tx, a hash value (denoted as e^f) is uniquely determined in the epoch when tx is disseminated by its client. Notably, e^f is unpredictable by adversaries, making them unable to predict the recipients of tx before e^f is public. To mitigate shortcoming (iii), an array function [158, 174] is first adopted by π^{τ} to randomly select τ replicas based on the value e^f. Then, the function generates an array, in which the τ replicas' indices are orderly listed. Finally, to reduce replicas' memory overhead, tx is sent only to the τ replicas ($\tau < f$).

Second, this chapter proposes Transaction Batch Generation (TBG), a π^{τ}-based batch-generation protocol for addressing the scheduling weakness in shortcoming (ii). TBG delays batch eligibility: a transaction *tx* sent to the Leaderless Byzantine Fault Tolerance (LBFT) network at epoch *e* can be selected for broadcast only from epoch $e + \kappa$, where κ is the network-skew parameter. TBG then applies the grouping-strided construction in Figure 4-1(d). Each replica first uses π^{τ} to place received valid transactions into τ buckets and then partitions these buckets into *G* groups. This schedule gives a transaction up to *G* broadcast opportunities while allowing later groups to observe whether an earlier group has already BC-delivered the transaction. As a result, TBG bounds post-delivery redundant broadcasts (RB) to one replica group.

Theorems 4.2 and 4.4 quantify the corresponding RB bound and transaction censorship (TC) rate.

We define $\mathcal{M}^{AS}$, a network model which is more restricted than the standard asynchronous network model [150]; we define $\mathcal{M}^{RC}$, a recovery-based corruption model in which a mobile adversary can continue corrupting replicas; we define two attacks to exacerbate the issues of RB and TC. We evaluate that under the HSM-based LBFT protocols, the TC rate can reach 76% under the above conditions. By replacing HSM with TBG, the simulated LBFT protocols obtain low RB and near-zero TC rate under these models and attacks. Moreover, under the assumptions stated in Section 4.2, we prove that TBG does not modify the batch broadcast or block agreement primitives of the scoped LBFT protocols. The contributions of this paper are listed as follows:

1. We analyse the TC rate of LBFT protocol under the above network model, corruption model, and two attacks.

2. We introduce π^{τ}, a protocol that counters Shortcomings (i) and (iii) and mitigates the issue of TC.

3. We propose TBG, which introduces a waiting period to resolve shortcoming (ii) and employs a grouping-strided construction to mitigate the issues of RB and TC.

4. Our simulation results show that the rate of TC of TBG-based LBFT protocol is near zero even when $\tau \ll f$, and that each transaction is broadcast, on average, three times under the above conditions. Our simulation results also show that the TBG-based protocol outperforms the HSM-based protocol in terms of admitted-transaction throughput, simulated latency, and per-replica memory overhead.

The remainder of this Chapter is organised as follows. Sections 4.2 and 4.3 introduce necessary background and system models the LBFT protocols rely on. Sections 4.4 and 4.5 detail the proposed protocols π^{τ} and TBG. Section 4.6 presents the simulation results. Section 4.7 concludes the Chapter.

4.2 Background

We consider a LBFT system that progresses in discrete epochs, with the e-th epoch indexed by e ($e \in \mathbb{N}$). Two primary participants exist: N replicas and C clients. Replicas are indexed by n ($n \in [N]$), where $[N]$ is short for $\{1,2,..,N\}$. Clients are indexed by c ($c \in [C]$). There is an adversary $\mathcal{A}$ who possesses finite resources and can control a fraction of participants. Participants controlled by $\mathcal{A}$, known as Byzantine participants, may behave arbitrarily, whereas those not under $\mathcal{A}$'s control are correct and adhere to protocol rules. We consider the adversary controls a proportion of ρ^C clients of the *C* clients, while the controlled replicas are defined in the recovery-based corruption mode. Communication between participants is over authenticated channel link.

4.2.1 Atomic Broadcast Protocol

We consider an atomic broadcast LBFT protocol P^{ABC}. Many asynchronous LBFT protocols – both ACS-based (e.g., HoneyBadgerBFT [141], Dumbo [142]) and DAG-based (e.g., Shoal [161]) – fit the following two-layer structure in each epoch e:

Batch Broadcast [175]*:* Each replica broadcasts one batch through a broadcast primitive (e.g., variants of reliable broadcast (RBC) [141] or provable broadcast (PB) [142]).

Block agreement: An agreement primitive (e.g., Asynchronous Binary Agreement (ABA) [141] and Multi-Value Byzantine Agreement (MVBA)) [142]) establishes a total order over batches. In DAG-based LBFT protocols, the block (vertices) agreement primitive is then replaced by a total-order algorithm which is locally executed by replicas [161].

Certificate: A certificate of a batch refers to a *batch recovery proof* under LBFT protocols [142]: if a correct replica obtains a valid certificate of a batch, then the batch is recoverable by every correct replica. The certificate can be a threshold signature [144]

or others [154]. With the certificate, LBFT protocols can develop the *strong validity* property [142] to defend against TC. P^{ABC} considers a certificate-based *three-message-flow* batch broadcast primitive:

Three-message-flow: In this Chapter, "three-message-flow" refers to a three-round communication of a certificate-based batch broadcast primitive: (F1) Batch dissemination (a replica broadcasts its batch or the dispersal metadata of its batch); (F2) Share dissemination (replicas multicast signed echoes/votes/shares, e.g., a threshold signature share, which are building blocks to generate a certificate); and (F3) Certificate dissemination (once a correct replica obtains a valid certificate, it broadcasts the certificate).

We assume that the considered P^{ABC} has the following four properties under the defined LBFT system:

Agreement: If a correct replica ABC-delivers tx_c, then every correct replica eventually ABC-delivers tx_c.

Integrity: For tx_c, every correct replica ABC-delivers tx_c at most once. Moreover, if a correct replica ABC-delivers tx_c with replica n, then tx_c was previously broadcast by replica n.

Strong Validity: If there exists a valid certificate for a batch such that a correct replica has broadcast it, then every correct replica eventually BC-delivers the batch and ABC-delivers the transactions (generated by correct clients) in the batch.

Total order: If a correct replica ABC-delivers tx_a before ABC-delivering tx_b, then no correct replica ABC-delivers tx_b without first ABC-delivering tx_a ($a \neq b$, $a \in [C]$, $b \in [C]$).

Specifically, ABC-delivery of tx_c by a replica means tx_c is orderly included in the ledger of the replica; BC-delivery of a batch means the batch has been treated as valid (but not ABC-delivered) by sufficient replicas (e.g., 2*f*+1 replicas). We further define a property regarding TC:

γ-censorship resistance. If correct client c disseminates its transaction tx_c to τ distinct replicas ($\tau < N$), tx_c is eventually ABC-delivered by all correct replicas with

a probability of γ.

In summary, this paper focuses only on P^{ABC} who preserves strong validity property under standard static or adaptive corruption models, as they are more robust to defend against TC compared to the protocols without the property.

Scoped LBFT protocol P^{ABC}. P^{ABC} is scoped by three conditions. First, if the same transaction appears in more than one ABC-delivered batch, all correct replicas apply the same deterministic rule and keep only the occurrence in the earliest ABC-delivered batch. Second, once a correct replica BC-delivers a batch certificate, the certificate and the corresponding batch are stored persistently so that a recovering replica can retrieve them later. Third, the protocol can progress in an epoch even when a correct replica has no eligible transaction to include in its batch, e.g., by broadcasting an empty batch. Notably, the three conditions have already achieved by corresponding strategies without modifying the batch broadcat and block agreement primitives.

Hash-space method (HSM). Before the execution of batch broadcast primitive, each replica needs to receive transactions from clients and select transactions to generate their batches. HSM is the protocol designed for transaction management and selection. HSM mitigates RB in P^{ABC} by partitioning a replica's memory into *N* hash space-based buckets. Buckets are deterministically managed by replicas in a rotating manner (e.g., replica 1 manages bucket 1 in epoch 1 and bucket 2 in epoch 2). A client maps a transaction to a bucket indexed by BID using its identifier ID and a timestamp ts: $BID = H(ID|ts) \bmod N$, where $H(\cdot)$ is a hash function and "|" is a concatenation operation. Notably, ts is non-deterministic, and a client can choose its own ts from a specific time range. We consider that HSM integrates the random selection method: in an epoch, if a replica's managed bucket contains more than *B* transactions, the replica randomly selects *B* transactions from the bucket, which further mitigates the issue of RB.

4.3 System Model

This section first articulates the proposed restricted network model, followed by the recovery-based corruption model. Finally, we define the transaction model – a crucial design for the proposed protocol π^{τ} to detect invalid transactions.

4.3.1 Restricted Network Model

To analyse scheduling attacks on LBFT batch generation, this section defines the restricted network model $\mathcal{M}^{AS}$. The standard asynchronous network model permits arbitrary finite message delays [150]. $\mathcal{M}^{AS}$ is more restricted: epoch duration remains unpredictable, but the scheduler is constrained by local-epoch skew and message delay. At global time *t*, let $e_p = e_p(t)$ be the local epoch index of a correct participant p. Here, $p \in [C]$ when the participant is a client, and $p \in [N]$ when the participant is a replica. Let $e_g = e_g(t)$ be the global epoch index, defined as the highest epoch index reached by at least one correct participant at time *t*. The functions $e_p(\cdot)$ and $e_g(\cdot)$ are monotone non-decreasing. The local epoch e_p means that p has seen the ABC-delivered block from epoch e_p-1 but has not yet seen the ABC-delivered block from epoch e_p. $\mathcal{M}^{AS}$ introduces κ to bound the distance between local and global epoch indices:

$$0 \leq e^{g}(t) - e_p(t) \leq \kappa - 1, \kappa = \{1,2\} \tag{4.1}$$

The constraint Eq. (4.1) stipulates that any correct participant cannot lag behind the fastest one by more than one epoch.

Let a correct participant p send a message m to a correct participant q at global time t_s. Let q first buffers m in its memory at global time t_r; $\mathcal{M}^{AS}$ then introduces the message-based "rubber-band" constraint:

$$0 \leq \left|e_q(t_r) - e_p(t_s)\right| \leq \kappa - 1, t_r > t_s \tag{4.2}$$

The constraint Eq. (4.2) mandates that message delivery between correct participants is delayed by at most one epoch, even if the epoch duration is unpredictable

(i.e., $t_r - t_s$ is unknown).

The parameter κ is the network-skew bound used in Eq. (4.1) and Eq. (4.2). It limits how far a correct participant's local epoch and buffered messages can lag behind the global epoch. These constraints are used only for the censorship analysis of batch generation. The agreement, integrity, strong validity and total-order properties are inherited from P^{ABC}. Because participants observe only local epochs, this chapter uses "local epoch" whenever a statement is made from a participant's perspective. Otherwise, "epoch" refers to the global epoch.

4.3.2 Recovery-based Corruption Model

Classical proactive and intrusion-tolerant systems extend leader-based state machine replication (e.g., PBFT) with periodic recovery or rejuvenation of replicas, so that a mobile adversary can compromise different nodes over time while controlling at most f replicas at any moment [168-173]. However, these works solely target leader-based BFT protocols and focus on system-wide safety and liveness, rather than on fine-grained transaction censorship in asynchronous LBFT protocols.

To capture dynamic adversarial behaviours and the ability of replica recovery in asynchronous LBFT protocols, we define a recovery-based corruption model $\mathcal{M}^{RC}$ tailored to the above epoch structure and the asynchronous network model $\mathcal{M}^{AS}$:

$\boldsymbol{\mathcal{M}^{RC}}$**.** Under the network model $\mathcal{M}^{AS}$, at any global time t, there are two time-metrics, i.e., global epoch and local epoch. To avoid ambiguity, $\mathcal{M}^{RC}$ introduces three definitions:

- A replica has only one state at any time t. For instance, if replica n $(n \in [N])$ is correct at global epoch $e^g(t)$, replica n is also correct at epoch $e_n(t)$, and vice versa.
- In each epoch, a replica has only one state.
- A mobile adversary $\mathcal{A}$ always perceives the global epoch, i.e., $e^g = e^g(t)$ at any global time t.

For N replicas in the system, $\mathcal{A}$ has two kinds of power:

Static Byzantine set: At initialisation of P^{ABC}, $\mathcal{A}$ permanently controls a fraction p^B of replicas. These replicas are Byzantine for the entire protocol execution.

Mobile Byzantine set: The rest of replicas form the *recoverable set* $\boldsymbol{S}^R$ of size $(1-p^B)N$. In each epoch e^g, $\mathcal{A}$ uses its finite resources to attack at most ℓ replicas from S^R. Attacked replicas do not become Byzantine instantly:

- If replica n is attacked at epoch e^g, it becomes actively Byzantine at the start of epoch $e^g + \lambda$, where λ is the corruption latency (measured in epochs);
- Upon becoming Byzantine at epoch $e^g + \lambda$, replica n remains under $\mathcal{A}$'s control during the epoch period in Eq. (4.3):

$$e^g + \lambda, e^g + \lambda + 1, \dots, e^g + \lambda + \lceil \varepsilon\lambda \rceil - \kappa - 1 \tag{4.3}$$

- From epoch $e^g + \lambda + \lceil \varepsilon\lambda \rceil - \kappa$ onwards ($\lceil \varepsilon\lambda \rceil > \kappa$), replica n is no longer controlled by $\mathcal{A}$ and enters a recovery phase. In this phase, replica n is temporarily crashed [177-179], but can fetch missing state from other replicas for state recovery. At epoch $e^g + \lambda + \lceil \varepsilon\lambda \rceil$, replica n becomes correct.

Resilient constraint: To ensure that $\mathcal{M}^{RC}$ is consistent with the standard BFT resilience bound, $\mathcal{M}^{RC}$ sets that:

$$p^B N + \lceil \varepsilon\lambda \rceil \ell \leq f, N = 3f + 1, 3p^B < 1 \tag{4.4}$$

Clarification on recovery overhead. In the recovery-based corruption model $\mathcal{M}^{RC}$, state recovery is represented as a crashed interval for the previously attacked replica. During this interval, normal scoped LBFT execution proceeds using batches and messages from replicas that Eq. (4.4) counts as correct. The recovering replica remains locally unavailable and contributes no batch, certificate share or block-agreement message. Other correct replicas continue to execute because Eq. (4.4) keeps the total number of Byzantine and crashed replicas within the bound *f*. Recovery traffic fetches missed certificates and batches for the recovering replica, but this traffic is outside the message sequence required for other correct replicas to deliver a block.

Within $\mathcal{M}^{RC}$, the model-level recovery overhead is κ epochs of local unavailability for that replica. If congestion or a large stored state extends recovery, the deployment should treat the replica as crashed for more epochs and re-check Eq. (4.4). The exact wall-clock recovery time depends on state-transfer implementation, stored-batch size, certificate format, disk layout and bandwidth; these implementation choices are outside the TC analysis in this chapter.

Each parameter in $\mathcal{M}^{RC}$ has a specific role in Eq. (4.4) and in the TC analysis [180, 181]. The parameter p^B represents $\mathcal{A}$'s capability to deploy its own Byzantine replicas into the LBFT system. ℓ reflects $\mathcal{A}$'s limited resources for attacking replicas per epoch, as $\mathcal{A}$ can attack at most ℓ replicas per epoch. λ models the inherent delay required to successfully corrupt replicas. ε further measures the temporary nature of Byzantine replicas from $\mathbf{S}^R$, as they can fully recover after $\lceil \varepsilon\lambda \rceil$ epochs. With these parameters, Theorem 4.1 gives the resilience property used in the later analysis.

Theorem 4.1. *Under the network model $\mathcal{M}^{AS}$, the model $\mathcal{M}^{RC}$ and the LBFT protocol P^{ABC}, at any global time point t, the number of incorrect replicas is no larger than f. Moreover, let a replica become correct at epoch $e^g + \lambda + \lceil \varepsilon\lambda \rceil$, it can recover the entire state of P^{ABC} at that epoch.*

Proof. At epoch e^g, the replicas attacked at the epochs $e^g - \lambda$, $e^g - \lambda - 1$, …, $e^g - \lambda - \lceil \varepsilon\lambda \rceil + \kappa + 1$ are Byzantine; the replicas attacked at the epochs $e^g - \lambda - \lceil \varepsilon\lambda \rceil + \kappa$, …, $e^g - \lambda - \lceil \varepsilon\lambda \rceil + 1$ are crashed. In the worst case ($e^{gl} \geq \lambda + \lceil \varepsilon\lambda \rceil$, $\mathcal{A}$ controls (i) $p^B N$ permanently Byzantine replicas, (ii) $(\lceil \varepsilon\lambda \rceil - \kappa)\ell$ Byzantine replicas and (iii) $\kappa\ell$ crashed replicas. According to Eq. (4.4), since $p^B N + (\lceil \varepsilon\lambda \rceil - \kappa)\ell + \kappa\ell \leq f$, the number of Byzantine and crashed replicas is no larger than f.

Let replica n become crashed at epoch $e^g + \lambda + \lceil \varepsilon\lambda \rceil - \kappa$ and request missing state of P^{ABC} from other replicas at this epoch. According to Eq. (4.2), every correct replica will receive replica n's requests and send the missing data to replica n before the end of epoch $e^g + \lambda + \lceil \varepsilon\lambda \rceil - \kappa + (\kappa - 1)$. Then, replica n will receive all the missing data before the end of $e^g + \lambda + \lceil \varepsilon\lambda \rceil - \kappa + 2(\kappa - 1)$. Since $\kappa \leq 2$, $e^g +$

$\lambda + \lceil \varepsilon\lambda \rceil - \kappa + 2(\kappa - 1) \leq e^g + \lambda + \lceil \varepsilon\lambda \rceil$. Thus, Theorem 1 is proven. ■

Theorem 4.1 ensures that the number of Byzantine or crashed replicas never exceeds the standard bound f at any time point. As a result, even if the recovery time for a crashed replica may be long, the model design ensures that the number of correct replicas is still larger than the required threshold f, making the LBFT protocol secure to be executed without crashed replicas.

4.3.3 Transaction Model

A transaction tx_c generated by a client indexed by c ($c \in [C]$) is expressed by a 6-tuple data structure:

$$tx_c = \{e_c, PubK_c, Req_c, Sig_{c,n}, DN_c, e_c^{new}\} \tag{4.5}$$

where e_c means that tx_c is disseminated at the client c's local epoch indexed by $e_c = e_c(t)$ (t is the global time when tx_c is disseminated by client c), $PubK_c$ is the public key of client c, which is available to all participants of the system, Req_c denotes the request of client c, $Sig_{c,n}$ denotes a signature, DN_c, initiated at 0, is a delivery number of client c, representing the number of client c's transactions that have been ABC-delivered by P^{ABC}, and e_c^{new} ($e_c^{new} > e_c$) is a future epoch index, which indicates that client c intends to send a new transaction no earlier than its local epoch indexed by e_c^{new}. Notably, a transaction produced in epoch *e* can be broadcast no earlier than *e*+1.

A transaction is valid if it satisfies the four following rules:

Rule-1 (Signature and contents correctness): When tx_c is sent to replica n, $Sig_{c,n}$ and Req_c must be valid based on the rules defined by P^{ABC} (or its involved applications [182]).

Rule-2 (Future-epoch constraint): Let tx_c^{last} be the last ABC-delivered transaction of client c with its future epoch index $e_c^{lastnew}$. tx_c could only be sent by client c after its local epoch indexed by $e_c^{lastnew} - 1$, i.e., $e_c^{new} > e_c \geq e_c^{lastnew}$.

Rule-3 (Monotone delivery number): For client c, DN_c of tx_c starts at 0 and is

incremented by 1 each time a transaction of client c is ABC-delivered by P^{ABC} (see Section 4.2.1).

Rule-4 (Per-epoch transaction bound): Let $\theta \in (0,1)$ be a system parameter chosen according to the batch size B and client number C. For client c ($\forall c \in [C]$), define:

$$\text{Elig}_c(e_c) = 1, \text{iff } H(PubK_c|e_c) < \theta \cdot 2^k \tag{4.6}$$

Then, tx_c cannot be submitted when $\text{Elig}_c(e_c) \neq 1$.

Rule-1 identifies incorrect contents of transactions. Since the validity of contents often depends on application scenarios [182], the definition of contents' validity is out of this paper's scope. Rule 2 implies that a client can only disseminate one transaction per epoch. Rule 3 accounts for the number of transactions have been ABC-delivered for a client. Rules 2 and 4 collectively limit the transaction loads per epoch.

4.3.4 Threat Model

We introduce two basic attacks, i.e., transaction flooding (TF) and timestamp malleability (TM). The TF attack leverages transactions that violate at least one of the transaction model's rules to saturate correct replicas' memory. It is achieved by the adversary $\mathcal{A}$ who generates up to $\rho^C C$ such transactions per epoch via its controlled $\rho^C C$ clients. In the TM attack, denoting tx^{inv} as one such transaction, $\mathcal{A}$ crafts N distinct timestamps for tx^{inv}, mapping variants of tx^{inv} to N different hash spaces, where N is the replica number.

Delay Flooding Epoch Scheduling (DFES) Attack. The DFES attack exacerbates the issue of RB under HSM. The DFES attack consists of three attack strategies:

Delaying batch: Based on HSM, at its local epoch e_n-1, replica n ($\forall n \in [N]$) broadcasts a batch of transactions selected from its hash space k. The adversary $\mathcal{A}$ makes replica n only able to generate the certificate for the batch at its local epoch $e_n - 1 + \Delta$, where $\Delta = 2\kappa - 2$ according to Eq. (4.2) and the three-message-flow-based broadcast primitive (see Section 4.2.1).

This strategy is the key to cause Shortcoming (ii). For instance, let replica m manage hash space k at epoch e_n. Since it cannot receive replica n's batch containing hash space k's transactions, replica m may rebroadcast these transactions.

Flooding invalid transactions: Consider the transactions that violate the rules defined in the transaction model as invalid transactions. Then, the strategy consists of two steps:

Step-1: $\mathcal{A}$ first uses the TF attack to generate $\rho^C C$ unique invalid transactions in epoch e_n-1. $\mathcal{A}$ then uses the TM attack to map the $\rho^C C$ transactions to the hash spaces that the correct replicas manage at epoch e_n (see Shortcoming (i)). For instance, if replica m manages bucket k at epoch e_n, $\mathcal{A}$ maps the $\rho^C C$ transactions to replica m's bucket k at epoch e_n-1.

Step-2: $\mathcal{A}$ makes the correct replicas in epoch e_n to generate and broadcast their epoch e_n's batches.

Step-1 of this strategy causes Shortcoming (iii), while step-2 of this strategy causes Shortcomings (i).

Epoch scheduling: The strategy also consists of two steps:

Step-3: $\mathcal{A}$ divides the $\rho^C C$ transactions into two groups: the first group contains the transactions that are included in the batches broadcast in step-2; the second group contains the transactions that are not in these batches broadcast in step-2.

Step-4: Based on Eq. (4.1), $\mathcal{A}$ stalls correct replica n at its own local epoch e_n-1. Then, $\mathcal{A}$ uses the TM attack to map the first group's transactions to hash space i that replica n manages at its local epoch e_n; $\mathcal{A}$ makes replica n proceed to epoch e_n and maps the second group's transactions to hash space j that replica n manages at its local epoch e_n+1.

This strategy causes shortcoming (ii), making a slow replica re-broadcast the transactions that have been broadcast.

Delay flooding epoch scheduling attack: Based on the three strategies, the DFES attack is generalised as follows:

At the end of an arbitrary epoch e-1, $\mathcal{A}$ selects a slow set (denoted as $\boldsymbol{S}_e^S$) of

$f' \leq f - n_e^C$ replicas who are correct at epoch e, where n_e^C is the number of crashed replicas at epoch e. $\mathcal{A}$ first delays their batches broadcast in epoch e via the strategy of delaying batch. The strategy of flooding invalid transactions is then applied to all replicas who are not in $\boldsymbol{S}_e^S$ and correct at epoch e+1. Finally, the epoch scheduling strategy is applied to all replicas who are in $\boldsymbol{S}_e^S$.

Random-Specific-Saturation (RSS) Attack. The RSS attack exacerbates the issue of TC. Let the targeted transaction tx^{tar} be sent to a set A^τ of τ distinct replicas at time point t, i.e., epoch $e^{tar} = e^g(t)$. Let tx^{tar} be allocated to hash space b, the RSS attack consists of two strategies:

Random attack: Before epoch e^{tar}, since $\mathcal{A}$ does not know the set A^τ of tx^{tar}, $\mathcal{A}$ randomly attacks ℓ different replicas in each of the epochs 1, …, $e^{tar} - \lambda + 1$, …, $e^{tar} - 1$. Based on the corruption model $\mathcal{M}^{RC}$, the replicas attacked at epoch e become Byzantine at the epoch $e + \lambda$, $e \in [e^{tar} - \lambda + 1, e^{tar} - 1]$.

Specific attack: At epoch e^{tar}, $\mathcal{A}$ divides the remaining correct and crashed replicas in A^τ into three groups. The first group includes the replicas that have been attacked during epoch e, $e \in [e^{tar} - \lambda + 1, e^{tar} - 1]$. The second group includes the replicas that are not attacked at epoch e but responsible for processing hash space j during their own local epochs $e^{tar} + \kappa \sim e^{tar} + \lambda - 1$. The third group includes the replicas that are not attacked at epoch e but responsible for processing hash space j after their local epoch $e^{tar} + \lambda - 1$.

$\mathcal{A}$ orders the replicas in the third group in ascending order of the epoch indices where they process hash space j. Then, in epoch e^{tar}, $\mathcal{A}$ attacks the first ℓ replicas in the third group; in epoch $e^{tar} + 1$, $\mathcal{A}$ attacks the second ℓ replicas in the third group; the logic continues, until all replicas in the third group are attacked. Afterwards, in each of the subsequent epochs, $\mathcal{A}$ randomly attacks ℓ replicas in the second group, until all of them are attacked. For any Byzantine replicas who hold tx^{tar}, $\mathcal{A}$ deletes tx^{tar} from their memory.

The two attack strategies ensure that each of the τ replicas holding tx^{tar} will eventually delete tx^{tar}.

Random-Specific-Saturation (RSS) Attack: Based on the two attack strategies, the RSS attack is designed as follows:

At epoch e^{tar}, $\mathcal{A}$ delays the dissemination of tx^{tar}, ensuring that all correct or crashed replicas in A^{τ} receive tx^{tar} at their own local epoch $e^{tar} + \kappa - 1$. Now, we consider that an arbitrary replica n ($n \in [N]$) from the first or second group is correct at its local epoch $e^{hsj} \in [e^{tar} + \kappa, e^{tar} + \lambda)$, while it manages hash space j at this epoch.

Strategy-1: According to Eq. (4.1), $\mathcal{A}$ stalls replica n at its local epoch $e^{hsj} - 1$ while driving the global epoch forward to $e^{hsj} + \kappa - 2$.

Strategy-2: Based on Eq. (4.2), $\mathcal{A}$ launches the TM attack: it forces replica n to store every invalid transaction (generated during the interval $[e^{hsj} - \kappa + 1,\ e^{hsj} + \kappa - 2]$) into hash space *j*, while making replica n unable to BC-deliver hash space *j*' transactions that have already been BC-delivered during the same interval $[e^{hsj} - \kappa + 1,\ e^{hsj} + \kappa - 2]$.

Analysis. The RSS attack floods replica n's hash space *j* at its local epoch $e^{hsj} - 1$. Since replica n can only select up to B transactions from hash space *j* to form a batch at its local epoch e^{hsj}, the probability that replica n selects tx^{tar} at that epoch will be diminished when hash space *j* is flooded.

Evaluation. The validity property in the LBFT protocols states that if some correct replica broadcasts a transaction tx, then all correct replicas eventually ABC-deliver tx. However, under the recovery-based corruption model $\mathcal{M}^{RC}$, the adversary can exploit the RSS attack to invalidate this property. Under the random/specific attack strategies of the RSS attack, even if tx is broadcast by a correct replica at its local epoch e, the replica may become Byzantine before it receives sufficient signed echoes/votes/shares messages. Under this context, the replica cannot broadcast the certificate of batch that contains tx, making tx unable to be BC-delivered. Moreover, under $\mathcal{M}^{RC}$ and the RSS attack, the adversary can make tx unable to be broadcast even when tx is disseminated to τ ($\tau > f$) replicas. Below we test the probability (p^{utb}) that a targeted transaction tx^{tar} is unable to be broadcast.

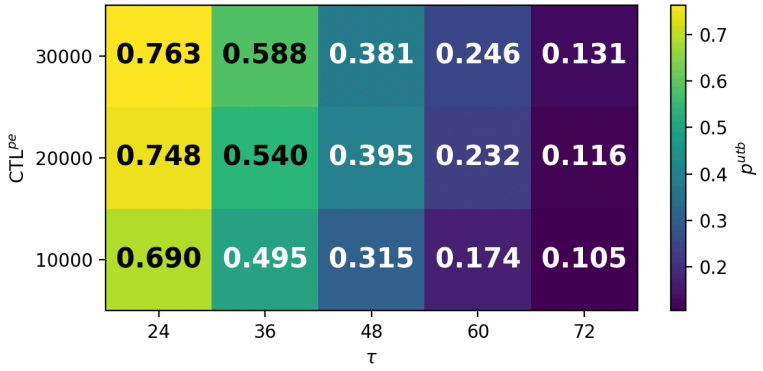

FIGURE 4-2. Evaluation on p^{utb} under different τ and CTL^{pe}.

The evaluation on p^{utb} operates within the HSM-based P^{ABC} under the intersection of the restricted network model $\mathcal{M}^{AS}$ and the recovery-based corruption model $\mathcal{M}^{RC}$. Let a targeted transaction mapped to a hash space j at epoch e. For simplicity purposes, we weaken the negative impact of memory saturation by excluding hash space j's transactions accumulated prior to epoch $e-1$. The replicas number N, the static control rate p^B (see Section 4.3.2), the block size B, the network parameter κ (see Eq. (4.1)), the corruption model parameters λ, ℓ and ε are set to be 100, 0.13, 2,500, 2, 10, 2, and 1, respectively. By setting τ to be {24, 36, 48, 60, 72} and CTL^{pe} (valid transaction load sent to the network per epoch) to be {10,000, 20,000, 30,000}, the results are presented in Figure 4-2.

It can be seen that p^{utb} exceeds 0.1 even at $\tau = 72$ and escalates sharply as the valid transaction load per epoch (CTL^{pe}) increases. This degradation is driven by the strategies 1 and 2 of the RSS attack. Specifically, the adversary floods the target replica's hash space j with an overwhelming volume of invalid transactions (approx. $2ITL^{pe}$) even under the simplified context. Consequently, even if a replica attempts to broadcast the targeted transaction tx^{tar}, the inclusion probability is drastically diluted: constrained by a fixed batch capacity B against a saturated hash space j of more than $2ITL^{pe}$ transactions, the likelihood of selecting tx^{tar} drops below $B/2ITL^{pe}$. Moreover, the RSS attack narrows the broadcast window: rather than all τ recipients acting as potential broadcasters, only a part of the τ replicas remain correct long enough to attempt the broadcast of tx^{tar} under the random/specific attacks of RSS.

4.4 Transaction Management

This section presents π^{τ} which identifies valid transactions and assigns each of them to τ different replicas for mitigating RB, defines undecided and invalid transactions, and removes invalid and BC-delivered transactions from replicas' memory.

4.4.1 Building Blocks of $\boldsymbol{\pi^{\tau}}$

π^{τ} adopts an array function $\mathcal{F}^{A}$ that orderly determines the τ recipients of a transaction. $\mathcal{F}^{A}$ accepts a nonce value n^{v} as an input, producing an array $\boldsymbol{A}^{\tau}$ containing τ orderly arranged integers (Figure 4-1(c)). $\mathcal{F}^{A}$ satisfies the two properties:

Uniformity: Prior to knowing the input of $\mathcal{F}^{A}$, every possible permutation of the τ integers in $\boldsymbol{A}^{\tau}$ is equal to $1/\tau!$.

Determinism: Once the input of $\mathcal{F}^{A}$ is fixed, the outputted array $\boldsymbol{A}^{\tau}$ and the order of the τ integers in $\boldsymbol{A}^{\tau}$ are determined.

π^{τ} relies on an unpredictable nonce value (denoted as n^{v}) to determine the τ recipients of a transaction tx_c generated by client c ($c \in [C]$), aiming to render adversaries unable to predict the τ recipients of tx_c before tx_c is disseminated. Specifically, client c retrieves its last ABC-delivered transaction tx_c^{last} which includes a future epoch index $e_c^{lastnew}$ (see Rule-2), a delivery number (denoted as DN_c^{Last}) and other four elements (see Eq. (4.5)). If tx_c^{Last} exists and client c's local epoch index e_c satisfies Rule-2 ($e_c \geq e_c^{lastnew}$) and Rule-4 ($\text{Elig}_c(e_c) = 1$), client c sets $DN_c = DN_c^{Last} + 1$ and obtains $h_c^{(e_c-1)}$ – the hash of the block ABC-delivered at epoch e_c-1; otherwise, client c needs to provide a proof (e.g., threshold signature [176]) for submitting its first transaction. Then,

$$n^{v} = DN_c | PubK_c | h_c^{(e_c-1)} \tag{4.7}$$

Once n^{v} of tx_c is determined, client c determines the τ replicas via the array

function $\mathcal{F}^A$.

4.4.2 τ-Partition Transaction Management Protocol π^{τ}

Let client c disseminate tx_c ($DN_c \geq 1$) to the τ replicas at its local epoch e_c. If replica n ($n \in [N]$) receives tx_c, it processes tx_c based on the procedures of π^{τ} (see Algorithm 4.1):

Replica n first checks whether Rule-1 of the transaction model is satisfied via the function $\mathcal{V}(\cdot)$ (lines 01-02). If not, tx_c is deemed invalid ($\mathcal{V}(tx_c) = 0$); otherwise, replica n uses $\mathcal{V}(\cdot)$ to check whether rules 2-4 are satisfied. According to Rule-2 (lines 03-04), (a) if tx_c^{last} is not ABC-delivered by replica n, tx_c is deemed undecided ($\mathcal{V}(tx_c) = \perp$); (b) if $e_c < e_c^{lastnew}$, tx_c is deemed invalid; (c) otherwise, there are three situations based on e_c and replica n's local epoch index e_n:

Situation 1 (line 05): $e_n < e_c$. If $e_c - e_n \geq \kappa$, tx_c is deemed invalid; otherwise, tx_c is deemed undecided. According to Eq. (4.1), no one can proceed to $e_n + \kappa$ even from the perspective of replica n. If $e_c - e_n \geq \kappa$, it means that client c launches an attack, as it does not proceed to e_c but disseminates tx_c that should be forwarded at e_c.

Situation 2 (lines 06-07): $e_n = e_c$. Replica n checks whether Rules 3 and 4 are satisfied. If either $DN_c^{(e_n)} = DN_c$ (see Rule-3) or $\text{Elig}_c(e) = 1$ (see Eq. (4.6) of Rule-4) is not satisfied, tx_c is invalid; otherwise, tx_c is valid ($\mathcal{V}(tx_c) = 1$), where $DN_c^{(e_n)}$ ($DN_c^{(e_n)} = DN_c^{Last} + 1$) is the delivery number of client c from the perspective of replica n's local epoch e_n.

Situation 3 (lines 08-16): $e_n > e_c$. Replica n first checks whether Rule-4 is satisfied or not. If Rule-4 is not satisfied, tx_c is deemed invalid; If Rule-4 is satisfied, replica n checks whether Rule-3 is satisfied according to $DN_c^{(e_n)}$ and DN_c:

Condition-1: If $DN_c^{(e_n)} > DN_c$, there must be a transaction tx_c^{last} whose delivery number DN_c^{Last}=DN_c has been ABC-delivered by replica n. Thus, replica n deletes

tx_c.

Condition-2: If $DN_c^{(e_n)} = DN_c$, tx_c is deemed valid.

Condition-3: If $DN_c^{(e_n)} < DN_c$, Rule-3 is not satisfied and tx_c is deemed invalid.

Once the type of tx_c is determined, the following procedures are executed according to tx_c's type:

When tx_c is undecided, tx_c is temporarily stored in replica n's memory. Once replica n advances its local epoch so that tx_c^{last} is ABC-delivered or $e_n = e_c$, the operation defined in line 17 or line 18 is executed.

When tx_c is valid and $DN_c^{(e_n)} = DN_c$, replica n inputs the nonce value n^v of tx_c (see Eq. (4.7)) into the array function $\mathcal{F}^A$ to produce the array $\boldsymbol{A}_c^\tau$, and checks whether its index n is listed in $\boldsymbol{A}_c^\tau$. If n ranks at the t-th position of $\boldsymbol{A}_c^\tau$ ($t \in [\tau]$), it stores tx_c at the t-th bucket generated at its local epoch $e_c + \kappa - 1$ and the bucket is denoted as $\boldsymbol{B}_{t,n}^{(e_c+\kappa-1)}$; otherwise, it discards tx_c from its memory (lines 19 and 20). Notably, a bucket in π^τ is a specific memory area for caching transactions that are valid and ready to be broadcast. Undecided transactions are stored in other memory area. BC-(ABC-)delivered ones are stored in permanent storage.

When $\mathcal{V}(tx_c)$=0, replica n discards tx_c (line 21); when tx_c is broadcast or BC-(ABC-)delivered by replica n, while tx_c is in replica n's bucket, it deletes tx_c from the bucket (line 22).

Every transaction stored in buckets will be further handled by the protocol TBG in Section 4.5.

Replica n determines the types of tx_c using the function $\mathcal{V}(\cdot)$. $\mathcal{V}(\cdot)$ first checks whether Rule-1 of the transaction model is satisfied (lines 01-03). If it fails, $\mathcal{V}(tx_c) = 0$ and tx_c is deemed invalid; otherwise, $\mathcal{V}(tx_c)$ further checks whether rules 2-4 are satisfied. According to rule-2, since tx_c cannot be submitted before epoch $e_c^{lastnew} - 1$, there are three situations based on tx_c^{last}'s $e_c^{lastnew}$ and replica n's current epoch index e_n:

Situation 1 (line 04-05): $e_n < e_c^{lastnew}$. If $e_c^{lastnew} - e_n \geq \kappa$, Rule-2 is not satisfied and tx_c is deemed invalid ($\mathcal{V}(tx_c) = 0$); otherwise, Rule-2 is satisfied and tx_c is deemed undecided ($\mathcal{V}(tx_c) = \perp$).

Situation 2 (lines 06-07): $e_n = e_c^{lastnew}$. Replica n checks whether rules 3 and 4 are satisfied. If either $DN_c^{(e_n)} = DN_c$ (see Rule-3) or $\text{Elig}_c(e) = 1$ (see Eq. (4.5) of Rule-4) is not satisfied, tx_c is deemed invalid; otherwise, tx_c is deemed valid, i.e., ($\mathcal{V}(tx_c) = 1$), where $DN_c^{(e_n)}$ is the delivery number of client c from the perspective of replica n who is at epoch e_n.

Situation 3 (lines 08-16): $e_n > e_c^{lastnew}$. Replica first checks whether rule-4 is satisfied or not. If Eq. (4.5) is not satisfied, tx_c is deemed invalid; otherwise, three conditions occur according to the values of $DN_c^{(e_n)}$ and DN_c:

Condition-1: If $DN_c^{(e_n)} > DN_c$, there must be a transaction tx'_c, generated by client c with its current delivery number also being DN_c, has been included in a block delivered by replica n. Replica n compares the request in tx'_c with Req in tx_c (see Eq. (4.4)): if the two requests differ, tx_c is deemed invalid; otherwise, tx_c is deemed valid.

Condition-2: If $DN_c^{(e_n)} = DN_c$, tx_c is deemed valid.

Condition-3: If $DN_c^{(e_n)} < DN_c$, tx_c is deemed invalid.

Once the type of tx_c is determined, the following procedures are executed according to tx_c's type:

When tx_c is undecided, tx_c is stored in replica n's idle memory. Once e_n of replica n advances so that $e_n = e_c^{lastnew}$, the operations defined in line 17 are executed. When tx_c is valid and $DN_c^{(e_n)} = DN_c$, replica n inputs n^v of tx_c into $\mathcal{F}^A$ to produce $\boldsymbol{A}_c^{\tau}$, and checks whether its index n is listed in $\boldsymbol{A}_c^{\tau}$. If the index of replica n ranks at the t-th position of $\boldsymbol{A}_c^{\tau}$ ($t \in [\tau]$), replica n stores tx_c at the t-th bucket generated at epoch $e + \kappa - 1$, the memory bucket is denoted by $\boldsymbol{MB}_{t,n}^{(e+\kappa-1)}$;

Algorithm 4.1 Procedures of π^{τ}

Let $tx_c = \{e_c, PubK_c, Req_c, Sig_{c,n}, DN_c, e_c^{new}\}$ generated by client c ($c \in [C]$) sent to replica n ($n \in [N]$) at client c's local epoch e_c; Let replica n's local epoch index be e_n:

01 $\mathcal{V}(\cdot) \rightarrow rule1(tx_c)$ based on Rule-1:

02 If $rule1(tx_c)$ is false: $\mathcal{V}(tx_c) = 0$

03 Else:

(a) if tx_c^{last} is not ABC-delivered: $\mathcal{V}(tx_c) = \perp$

04 (b) if $e_c < e_c^{lastnew}$, $\mathcal{V}(tx_c) = 0$;

(c) if $e_c \geq e_c^{lastnew}$: turn to the three situations

Situation 1: $e_n < e_c$

05 If $e_c - e_n \geq \kappa$: $\mathcal{V}(tx_c) = 0$; Else: $\mathcal{V}(tx_c) = \perp$

Situation 2: $e_n = e_c$

06 If $DN_c^{(e_n)} \neq DN_c$ and/or $\text{Elig}_c(e_c) \neq 1$: $\mathcal{V}(tx_c) = 0$

07 Else: $\mathcal{V}(tx_c) = 1$

Situation 3: $e_n > e_c$

08 If $\text{Elig}_c(e_c) \neq 1$: $\mathcal{V}(tx_c) = 0$

09 Else:

10 Condition 1: If $DN_c^{(e_n)} > DN_c$: execute line 22

11 If $Req_c^{last} = Req_c$: $\mathcal{V}(tx_c) = 1$

12 Else: $\mathcal{V}(tx_c) = 0$

13 Condition 2: If $DN_c^{(e_n)} = DN_c$:

14 $\mathcal{V}(tx_c) = 1$

15 Condition 3: If $DN_c^{(e_n)} < DN_c$:

16 $\mathcal{V}(tx_c) = 0$

When $\mathcal{V}(tx_c) = \perp$

17 Once tx_c^{last} is ABC-delivered: execute lines 05

18 Once $e_n = e_c$: execute lines 06-07

When $\mathcal{V}(tx_c) = 1$ and $DN_c^{(e_n)} = DN_c$

19 compute $\mathcal{F}^A(DN_c|PubK_c|h_c^{(e_c-1)})$

20 If $n \in A_c^{\tau}$: store tx_c in $B_{t,n}^{(e_c+\kappa-1)}$; Else, discard tx_c

When $\mathcal{V}(tx_c) = 0$

21 Discard tx_c from its memory

When tx_c is broadcast or BC-(ABC-)delivered

22 Delete tx_c from the corresponding bucket

otherwise, replica n deletes tx_c (lines 18 and 19).

Finally, if tx_c is invalid, replica n deletes tx_c (line 20); if tx_c is in a delivered block, while tx_c is still in replica n's memory, replica n deletes tx_c (line 21).

4.4.3 Analysis of π^{τ}

π^{τ} addresses Shortcomings (i) and mitigates the issue of shortcomings (iii). As for Shortcoming (i) caused by the TM attack (see Section 4.3.4), π^{τ} uses the deterministic nonce value n^{v} to replace the non-deterministic timestamp (in HSM) for a transaction tx_c. Since n^{v} is a unique and deterministic value, a malicious client cannot generate multiple variants of tx_c using different n^{v}. Thus, the TM attack is resolved.

Shortcoming (iii) is mitigated via three strategic designs. First, for tx_c's array $\boldsymbol{A}^{\tau}$, if index n ($n \in [N]$) is not in $\boldsymbol{A}^{\tau}$, replica n will discard tx_c. Second, it is trivial to prove that π^{τ} can detect and eliminate invalid transactions based on the 4-rule of the transaction model, which reduces the bucket-based memory overhead caused by the flooding invalid transactions strategy (see Section 4.3.4). Third, following Rules 2 and 4, the valid transaction load per epoch (denoted as CTL^{pe}) is limited to alleviate the bucket-based memory overhead.

By reducing the bucket-based memory overhead of correct replicas, the transaction loads of correct replicas' buckets are largely reduced, indicating that valid transactions have more chance to be broadcast, thus mitigating the issue of TC. Moreover, since the adversary $\mathcal{A}$ cannot know the exact e_c before client c submits tx_c (the unpredictable feature of n^{v}), $\mathcal{A}$ cannot launch the specific attack strategy (see Section 4.3.4) before tx_c is public, which also mitigates the issue of TC.

However, π^{τ} does not solve Shortcoming (ii), as the scheduler can still launch the scheduling attack specified in the DFES attack to cause RB. In Section 4.5, we introduce TBG to resolve the issue of RB without exacerbating the issue of TC.

4.5 Batch Generation

TBG assembles a replica's batch at the beginning of each epoch. It uses a waiting period and a grouping-strided design to reduce RB while preserving broadcast opportunities for valid transactions. Let client *c* disseminate transaction txc to replica *n* at local epoch ec. If replica *n* stores tx_c in the *t*-th bucket generated at local epoch $ec + \kappa - 1$, Algorithm 4.2 allows replica *n* to select tx_c only from local epoch $ec + \kappa$. This waiting period follows the κ-bound in Eq. (4.1) and Eq. (4.2), and it gives correct replicas time to observe whether an earlier group has already BC-delivered tx_c before another group rebroadcasts it.

Because π^{τ} sends txc to τ replicas, TBG partitions these τ holders into G groups. Replicas in the first group can broadcast txc at local epoch $ec + \kappa$. Replicas in the second group broadcast only if they have not BC-delivered txc before local epoch $ec + 3\kappa$. Later groups follow the same 2κ-strided schedule. For tx_c, the cycle ends when one correct replica BC-delivers tx_c or when the last scheduled local epoch $ec + 2\kappa G - \kappa$ is reached. This schedule gives at most *G* broadcast opportunities. The *G*-grouping design bounds broadcasts by group size $g = \lfloor \tau/G \rfloor$, as quantified in Theorem 4.2. The 2κ stride gives later groups time to observe certificates from earlier groups, as stated in Lemma 4.1.

Lemma 4.1. *Consider that* tx_c *is disseminated by client* c *at its local epoch* e_c. *For* tx_c *broadcast by some of the* x*-th group's replicas at their local epoch* $e_c + 2\kappa x - \kappa$ *(*$x \in [G]$*), the strided design ensures that every correct replica in the* $(x + 1)$*-th group can determine whether it has BC-delivered* tx_c *before the start of its local epoch* $e_c + 2\kappa x + \kappa$.

Proof. Under the three-message-flow broadcast primitive (see Section 4.2.1), if a correct replica n broadcasts a batch containing tx_c at its local epoch $e_c + 2\kappa x - \kappa$, every replica can receive the batch and send a reply-message of the batch to replica n before the end of their own local epoch $e_c + 2\kappa x - 1$ (see Eq. (4.2)). Then, if replica n is still correct at its local epoch $e_c + 2\kappa x + \kappa - 2$, it will receive sufficient reply-

messages to generate a certificate for the batch and broadcast the certificate at the epoch; otherwise, it will not broadcast the certificate. Finally, by judging whether they receive the certificate before their local epoch $e_c + 2\kappa x + 2\kappa - 3$, correct replicas can determine whether it has BC-delivered tx_c at the epoch. Since $\kappa = \{1,2\}$ (see Eq. (4.1)), $e_c + 2\kappa x + 2\kappa - 3 < e_c + 2\kappa x + \kappa$. ■

4.5.1 Details of TBG

As shown in Figure 4-3, in each of its local epochs, replica n ($n \in [N]$) partitions its generated τ buckets into G groups. Each of the first $G-1$ groups has g buckets; the last group has $\tau - (G-1)g$ buckets. At its local epoch e_n, there are two operations to generate a batch before initiating P^{ABC}. The pseudocode of TBG is presented in Algorithm 4.2.

Replica n invokes π^{τ} (lines 17 and 22 of Algorithm 4.1) to delete transactions in the τ buckets across G epochs, i.e., $\boldsymbol{B}_{1,n}^{(e_n-1)}$,…, $\boldsymbol{B}_{g,n}^{(e_n-1)}$, $\boldsymbol{B}_{g+1,n}^{(e_n-2\kappa-1)}$,…, $\boldsymbol{B}_{2g,n}^{(e_n-2\kappa-1)}$,…, $\boldsymbol{B}_{(G-1)g+1,n}^{(e_n-2\kappa(G-1)-1)}$,…, $\boldsymbol{B}_{\tau,n}^{(e_n-2\kappa(G-1)-1)}$, where $\boldsymbol{B}_{t,n}^{(e)}$ is the t-th bucket generated by replica n at its local epoch e ($t \in [\tau]$, $e = \{e_n - 2\kappa \times (G-1) - 1, \dots, e_n - 2\kappa \times 1 - 1, e_n - 2\kappa \times 0 - 1\}$). This operation ensures that invalid transactions and BC-(ABC-) delivered transactions will not be broadcast (line 01).

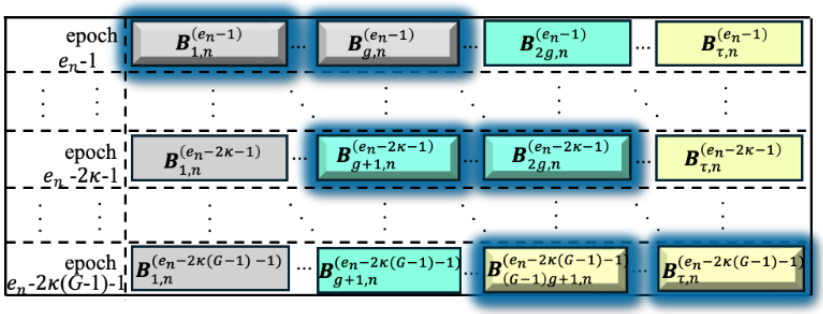

FIGURE 4-3. The diagram of TBG's batch assembly. In each epoch, replica n ($n \in [N]$) creates τ buckets and divides them into G groups (Gray boxes denote the g buckets of the first group; blue boxes denote the g buckets of the second group; and yellow boxes denote the remaining $\tau - g(G-1)$ buckets of the final group). During epoch e_n, replica n gathers transactions from the highlighted deep-blue (prism-shaped) buckets into its proposal batch.

Algorithm 4.2 Procedures of TBG

Replica n ($n \in [N]$) in its local epoch e_n:

01 Invoke π^{τ} to delete transactions from memory

02 Take transactions from $\boldsymbol{B}_{1,n}^{(e_n-1)}, \ldots, \boldsymbol{B}_{g,n}^{(e_n-1)}$; $\boldsymbol{B}_{g+1,n}^{(e_n-2\kappa-1)}, \ldots, \boldsymbol{B}_{2g,n}^{(e_n-2\kappa-1)}$; …; $\boldsymbol{B}_{(G-1)g+1,n}^{(e_n-2\kappa(G-1)-1)}, \ldots, \boldsymbol{B}_{\tau,n}^{(e_n-2\kappa(G-1)-1)}$

At local epoch e_n, replica n builds its batch from remaining transactions in selected historical buckets. The first group's g buckets come from the most recent eligible bucket set. Later groups use bucket sets generated at earlier 2κ-spaced local epochs. The G-th group contains the remaining $\tau - (G-1)g$ buckets. The exact bucket indices are listed in line 02 of Algorithm 4.2. The generated batch is then passed to P^{ABC}'s batch broadcast primitive.

4.5.2 Analysis of TBG

The analysis establishes three properties of TBG. Theorem 4.2 bounds RB under the DFES attack. Theorem 4.3 shows that TBG changes only batch contents and does not change PABC's batch broadcast primitive or block agreement primitive. Theorem 4.4 quantifies the TC rate of TBG-based PABC under the RSS attack.

Theorem 4.2. *Consider the three system models defined in Section 4.3.3 and the DFES attack defined in Section 4.3.4. Under the TBG-based P^{ABC}, a BC-delivered transaction can be broadcast by at most $\max\{g, \tau - g(G-1)\}$ times (g=$\lfloor \tau/G \rfloor$).*

Proof. The DFES attack (see Section 4.3.4) worsens the issue of RB via its three strategies, i.e., delaying batch, invalid transaction flooding, and epoch scheduling. Since π^{τ} eliminates invalid transactions, the flooding strategy is resolved. Then, based on Lemma 4.1, before a replica plans to broadcast a transaction tx, it can determine whether it has BC-delivered tx, which resolves the strategies of delaying batch and epoch scheduling. Moreover, with the grouping design and Lemma 4.1, tx BC-delivered by the x-th group's replicas cannot be broadcast by the correct replicas in the remaining groups. Thus, BC-delivered tx can only be broadcast by one group's replicas. ■

Theorem 4.3. *For the scoped P^{ABC}, after integrating TBG, P^{ABC}'s two primitives defined in Section 4.2.1 remain unchanged.*

Proof. P^{ABC} is modeled as an atomic-broadcast LBFT protocol which consists of a batch broadcast primitive and a block agreement primitive. According to the proofs of the state-of-the-art LBFT protocols, the four properties of P^{ABC} (see Section 4.2.1)

do not rely on batches' contents, but the designed instructions of the two primitives. Since TBG only determines the contents of batches, instead of changing any details of P^{ABC}'s two primitives, Theorem 4.3 is proven. ■

Theorem 4.4. *Consider the three models defined in Section 4.3 and the RSS attack defined in Section 4.3.4. At epoch e_c, let a valid targeted transaction tx^{tar} be disseminated to the τ replicas via π_τ ($e_c > \lambda + \lceil \varepsilon\lambda \rceil$ and $e_c = e_c = e^g$, the worst-case illustrated in Theorem 4.1). Let $\Delta = 2\kappa - 2$ and $G' \leq G$. Under the TBG-based P^{ABC} and Lemma 4.1, the expected γ-censorship resistance (denoted γ^{TBG}) is then given by:*

$$\gamma^{TBG} = 1 - \prod_{h=1}^{H}(1 - p^{\text{suc}}_{e_c+h+\Delta}) \tag{4.8a}$$

$$p^{\text{suc}}_{e_c+h+\Delta} = \begin{cases} 0, h \neq (2x-1)\kappa, x \in [1, G'] \\ 1 - (1 - \frac{1}{G'} \cdot \frac{n^{\tau}_{e_c+h+\Delta}}{n^{\tau}_{e_c+h}})^{n^{\tau}_{e_c+h}}, h = (2x-1)\kappa \end{cases} \tag{4.8b}$$

$$n^{\tau}_{e_c+h} = \begin{cases} \tau - (f - \kappa\ell)(\frac{\tau}{N}) - h\ell(\frac{\tau}{N}), 0 \leq h \leq \lambda - 1 \\ n^{\tau}_{e_c+\lambda-1} - (h - (\lambda-1))\ell, \lambda - 1 < h < H \\ 0, h \geq H \end{cases} \tag{4.8c}$$

$$n^{\tau}_{e_c+\lambda-1} = \tau - (f - \kappa\ell)(\frac{\tau}{N}) - (\lambda - 1)\ell(\frac{\tau}{N}) \tag{4.8d}$$

where $p^{\text{suc}}_{e_c+h+\Delta}$ is the expected probability that at least one correct replica BC-delivers tx^{tar} at epoch $e_c + h + \Delta$, H is the expected first epoch at which the expected number of correct holders of tx^{tar} becomes 0, $n^{\tau}_{e_c+h}$ is the expected number of correct replicas that hold tx^{tar} at epoch $e_c + h$.

Proof. Let tx^{tar} be the targeted transaction submitted to a set of τ replicas (denoted as $\boldsymbol{A}^{\tau}$) by client c at its local epoch e_c. Under the RSS attack specified in Section 4.3.4, we analyse the probability γ^{TBG} that tx^{tar} is BC-delivered by at least one correct replica in $\boldsymbol{A}^{\tau}$ before epoch $e_c + H$.

1. Derivation of n^{τ}_{e+h}. We compute $n^{\tau}_{e_c+h}$, the expected number of correct replicas who hold tx^{tar} at epoch $e_c + h$. To censor tx^{tar}, the adversary $\mathcal{A}$ launches the RSS attack:

- **Random Attack** ($0 \le h \le \lambda - 1$). Before tx^{tar} is disseminated, $\mathcal{A}$ randomly attacks ℓ replicas per epoch. Since $\mathcal{A}$ does not know $\boldsymbol{A}^{\tau}$ before epoch e_c, the probability that any replica (attacked before epoch e_c) lies in $\boldsymbol{A}^{\tau}$ is τ/N. According to the recovery-based corruption model $\mathcal{M}^{RC}$ and $e_c > \lambda + \lceil \varepsilon\lambda \rceil$, there are $p^B N + (\lceil \varepsilon\lambda \rceil - \kappa)\ell = (f - \kappa\ell)$ replicas that are Byzantine at epoch e_c. The expected number of these replicas in $\boldsymbol{A}^{\tau}$ is $(f - \kappa\ell)(\frac{\tau}{N})$. Over h epochs since epoch e_c, there are $h\ell$ replicas that become Byzantine during epochs $e_c + 1$ to $e_c + h$. Among them, the expected number of Byzantine replicas in $\boldsymbol{A}^{\tau}$ is $h\ell(\frac{\tau}{N})$. According to RSS, once a replica becomes Byzantine, if it receives tx^{tar}, it deletes tx^{tar}. Thus, the expected number of replicas who hold tx^{tar} at epoch $e_c + h$ (i.e., $n^{\tau}_{e_c+h}$) is $\tau - (f - \kappa\ell)(\frac{\tau}{N}) - h\ell(\frac{\tau}{N})$, when $h = \lambda - 1$, $n^{\tau}_{e_c+\lambda-1} = \tau - (f - \kappa\ell)(\frac{\tau}{N}) - (\lambda - 1)\ell(\frac{\tau}{N})$, which confirms Eq. (4.8d).
- **Specific Attack** ($\lambda \le h < H$). Once tx^{tar} is disseminated, $\mathcal{A}$ identifies the set $\boldsymbol{A}^{\tau}$ immediately. Thus, in epoch e_c, $\mathcal{A}$ strategically attacks ℓ replicas in $\boldsymbol{A}^{\tau}$ according to the specific attack strategy defined in RSS. The operation continues in epochs $e_c + 1$, …, until all $\boldsymbol{A}^{\tau}$'s correct and crashed replicas are attacked. Thus, according to the λ corruption latency defined in the corruption model $\mathcal{M}^{RC}$, in each of the epochs $e_c + \lambda$, …, $e_c + H - 1$, ℓ replicas in $\boldsymbol{A}^{\tau}$ become Byzantine.

Taken together, at epoch $e_c + h$, (i) when $h \le \lambda - 1$, the expected number of correct replicas who hold tx^{tar} is $\tau - (f - \kappa\ell)\left(\frac{\tau}{N}\right) - h\ell(\frac{\tau}{N})$; (ii) when $\lambda \le h < H$, the expected number of correct replicas who hold tx^{tar} is $n^{\tau}_{e_c+\lambda-1} - (h - (\lambda - 1))\ell$; when $h \ge H$, the expected number of correct replicas (in $\boldsymbol{A}^{\tau}$) who hold tx^{tar} is 0, which confirms Eq. (4.8c).

2. Derivation of $p^{suc}_{e_c+h+\Delta}$. We compute $p^{suc}_{e_c+h+\Delta}$, the expected probability that at

least one correct replica BC-delivers tx^{tar} at epoch $e_c + h + \Delta$. $p^{suc}_{e_c+h+\Delta}$ is influenced by two aspects: (a) since a replica can only broadcast B transactions per epoch, if the number of transactions (denoted as TL) to be broadcast is larger than B, $p^{suc}_{e_c+h+\Delta}$ may be less than 1. (b) under TBG, even if a correct replica buffers tx^{tar}, it allows the replica to broadcast tx^{tar} at a specified epoch (e.g., epoch $e_c + \kappa$ when it is allocated to the first group via π^{τ} and TBG).

As for aspect (a), the RSS attack proliferates TL via strategy-1 and strategy-2 (see Section 4.3.4). Specifically, if replica n is assigned to the x-th group, it will broadcast tx^{tar} at its local epoch $e_c + 2\kappa x - \kappa$. Then, the adversary $\mathcal{A}$ first leverages strategy-1, stalling replica n in its local epoch $e_c + 2\kappa x - \kappa - 1$. Then, $\mathcal{A}$ floods the valid and invalid transactions generated or BC-(ABC-) delivered during $[e_c + 2\kappa x - 2\kappa + 1,\ e_c + 2\kappa x - 2]$ into bucket j storing tx^{tar}, making TL significantly larger than B.

The RSS attack is resolved by the three designs:

- First, π^{τ} ensures that invalid transactions will be directly identified and deleted from memory. Hence, $\mathcal{A}$ cannot flood replica n's bucket j with invalid transactions.
- Second, Lemma 4.1 ensures that the correct replicas in the x-th group ($x \in [G']$) will not re-broadcast the BC-delivered transactions that are broadcast by some replicas in the $(x-1)$-th group. Thus, even if $\mathcal{A}$ floods replica n's bucket j with BC(ABC-) delivered valid transactions, replica n will not broadcast them (see line 01 of Algorithm 4.2).
- Third, by selecting a proper θ defined in Rule-4 of the transaction model, the total number of valid transactions stored in the τ buckets will not be larger than B.

Based on the three designs, as long as replica n is correct at its local epoch $e_c + 2\kappa x - \kappa$, it can broadcast the targeted transaction tx^{tar} at the epoch for sure. Thus, the impact of aspect (a) is resolved.

As for aspect (b), let $h = 2\kappa x - \kappa$ with $x \in [1, G']$. According to Eq. (4.8c), $n^{\tau}_{e_c+h}$ replicas from the set $\boldsymbol{A}^{\tau}$ remain correct at their own local epoch $e_c + h$.

(i) Let a replica from $\boldsymbol{A}^{\tau}$ be assigned to broadcast tx^{tar} at its local epoch $e_c + h$. According to Lemma 4.1, for the $(x+1)$-th group to determine whethertx^{tar} has been BC-delivered before its own scheduled epoch, the broadcasting replica must remain correct from its local epoch $e_c + h$ to $e_c + h + \Delta$ ($\Delta = 2\kappa - 2$). Under the RSS attack, the adversary continuously corrupts replicas. Given that the number of correct holders decreases from $n^{\tau}_{e_c+h}$ to $n^{\tau}_{e_c+h+\Delta}$ over Δ epochs, the conditional probability that a replica correct at $e_c + h$ remains correct at $e_c + h + \Delta$ is given by the ratio r_{surv}:

$$r_{surv} = \frac{n^{\tau}_{e_c+h+\Delta}}{n^{\tau}_{e_c+h}}, n^{\tau}_{e_c+h} > n^{\tau}_{e_c+h+\Delta} \geq 0$$

(ii) According to the *G*-based grouping design and the uniformity property of π^{τ}, under the RSS attack, the probability p_{sel} that a replica from the set $\boldsymbol{A}^{\tau}$ is allocated to the x-th group is $1/G'$.

Based on (i) and (ii), the joint probability that a specific correct replica successfully broadcasts tx^{tar} at its local epoch $e_c + h$ and remains correct at its local epoch $e_c + h + \Delta$ is:

$$p_1 = p_{sel} \cdot r_{surv} = \frac{1}{G'} \cdot \frac{n^{\tau}_{e_c+h+\Delta}}{n^{\tau}_{e_c+h}}$$

Let p_2 be the probability that no such correct replicas among the $n^{\tau}_{e_c+h}$ replicas exist in their local epoch $e_c + h$. Based on (i) and (ii), $p_2 = (1 - p_1)^{n^{\tau}_{e_c+h}}$. Thus, the probability that at least one such correct replica at its local epoch $e_c + h + \Delta$ is $p^{suc}_{e_c+h+\Delta} = 1 - p_2$.

3. Cumulative Censorship Resistance γ^{TBG}. Let E_h be the event that tx^{tar} is BC-delivered by a correct replica at its local epoch $e_c + h + \Delta$ for the first time. This event requires failure in all previous scheduled epochs $e_c + \kappa + \Delta$, $e_c + 3\kappa + \Delta$ …,

$e_c + h + \Delta - 2\kappa$ and success at $e_c + h + \Delta$, which leads to $P(E_h) = p^{suc}_{e_c+h+\Delta} \prod_{j=0}^{h+\Delta-2\kappa}(1 - p^{suc}_{e_c+j})$, $p^{suc}_{e_c+j} = 0$ when $j \neq \{\kappa + \Delta, 3\kappa + \Delta, \ldots\}$. Summing these mutually exclusive events over the window up to H confirms γ^{TBG} in Eq. (4.8a). ■

4.6 Simulation

In this section, evaluations on the TBG-based P^{ABC} and comparisons between TBG-based P^{ABC} and HSM-based P^{ABC} are conducted to show the performance of TBG-based P^{ABC}.

4.6.1 Simulation Setup

Our simulations run on a laptop with an Apple M2 processor and 16 GB of RAM and are fully reproducible using several thousand lines of Python code. In the simulation, we adopt the asynchronous protocol, called SpeedingDumbo [176], as the atomic broadcast protocol P^{ABC}, treating SpeedingDumbo and P^{ABC} interchangeably throughput the rest of this paper. As detailed in [176], P^{ABC} incurs 14 steps per epoch on average. We model the latency of each step (denoted as L^s) as:

$$L^s = P^d + L^q + L^c + L^{tx} + L^r \tag{4.9}$$

where $P^d = 160$ ms is the physical-channel propagation delay; $L^q = 5\mu s$ is per-message network-stack/queuing delay; $L^c = 20$ ms covers computation delays such as signature generation and verification; L^{tx} is the transmission delay with a 5 megabytes per second; and $L^r = 100$ms is a fixed retransmission penalty at a 0.1% packet-loss rate. All parameters above are held constant for reproducibility.

We integrate our proposed TBG and HSM (see Section 4.1.1) into P^{ABC} for replicas to assemble their batches at the beginning of each epoch, respectively. For comparison of the two protocols, the seven metrics are defined for this comparison:

Transaction censorship rate (R^{atc}): Over ce consecutive epochs starting at epoch

e, $CTL^{pe} \cdot ce$ unique valid transactions are submitted to the LBFT system. Let N^d be the number of these transactions delivered by the end of epoch $e + ce + \kappa G$. Then, $R^{atc} = 1 - \frac{N^d}{CTL^{pe} \cdot ce}$.

Average redundant broadcasts per transaction (C^{ARB}): For the N^d delivered valid transactions, let C^{RB} denote their total number of broadcasts. Then, $C^{ARB} = \frac{C^{RB} - N^d}{N^d}$.

Average throughput (Tps^A): Over ce consecutive epochs, let N^s be the total number of transactions delivered. Then, $Tps^A = \frac{N^s}{ce \cdot 14L^s}$.

Average latency (L^A): Consider L valid transactions - submitted in epoch e - have been delivered. For the l-th transaction among the L transactions ($l \in [L]$), let e_l denote its delivered epoch index. Then, $L^A = \frac{1}{L}\sum_{l=1}^{L}(e_l - e - 1)$.

Average memory overhead of a correct replica per epoch (AMC^{re}): Consider $N_n^{(e)}$ transactions be stored in replica n's memory ($n \in [N]$) at the end of epoch e. Then, $N_n^{(e)} = \frac{1}{N}\sum_{n=1}^{N} N_n^{(e)}$.

We consider the size of a transaction and a batch to be 300 bytes and 2,500, respectively. The attacks related to TF (see Section 4.3.4) are considered by setting $CTL^{pe} = \rho^C C$, meaning that the number of valid transactions sent to the system per epoch is equal to the number of invalid transactions sent to the system per epoch. Due to the waiting period design, we assume that the transaction generated in epoch e can be completely processed by π^τ before the end of epoch $e + \kappa - 1$. This assumption is justified as the most computationally intensive operations for processing a single transaction – specifically the array function $\mathcal{F}^A$, equation of Eq. (4.6), and signature verification – incur only microsecond-level latency, which is negligible compared to the typical epoch duration (e.g., second-level latency). Moreover, based on our local testing, the time for selecting transactions from buckets is only several microseconds, we thus omit this time cost introduced by TBG. Under the above settings, the evaluations are presented as below.

4.6.2 Evaluation on Transaction Censorship and Redundant Broadcast

To quantify the impact of the RSS attack in Section 4.3.4, this section evaluates the average censorship rate Ratc of TBG-based PABC for a target transaction txtar. The evaluations use the restricted network model MAS, the recovery-based corruption model MRC and the transaction model. The fixed parameters are $\kappa = 2, l = 2, \varepsilon = 1$, $p^B = 0.13$ and $Tr = 30{,}000$ independent trials. Figure 4-4 varies λ in the corruption model, τ in π^τ and G in TBG.

Figure 4-4 shows that increasing the ratio τ/G markedly lowers R^{atc}. The reason is that higher τ/G makes a group have more replicas, which increases the probability $p^{suc}_{e_c+h+\Delta}$ (see Eq. (4.8b)). In contrast, the influence of λ on R^{atc} is complex. Increasing λ from 8 to 10 can reduce R^{atc}. However, increasing λ from 6 to 8 will bring side effects. This complexity actually can be attested via analysing Eq. (4.8c), as λ plays both positive and negative roles on increasing $n^{\tau}_{e_c+h}$ and thus γ^{TBG}. Despite the complex influence of λ, R^{atc} is nearly zero when $\tau/G \geq 4$, which confirms the effectiveness of TBG for mitigating TC even under the RSS attack and a small τ.

Figure 4-4 also suggests a practical selection rule. After κ is fixed due to the restricted asynchronous network design, τ in π^τ should be selected together with G: the ratio τ/G determines the expected number of replicas in each group; a smaller G or a larger τ increases group size and the probability in Eq. (4.8b). A smaller G also increases the upper bound on RB in Theorem 4.2. A larger τ requires each client to disseminate a transaction to more replicas. A larger G reduces RB, but each group contains fewer replicas. Therefore, among configurations of applications that satisfy the target TC rate, the implementer should prefer the larger G and the smaller τ to mitigate RB and improve throughput, latency and memory overhead. Otherwise, the implementer should prefer larger τ to ensure that a transaction cannot be censored.

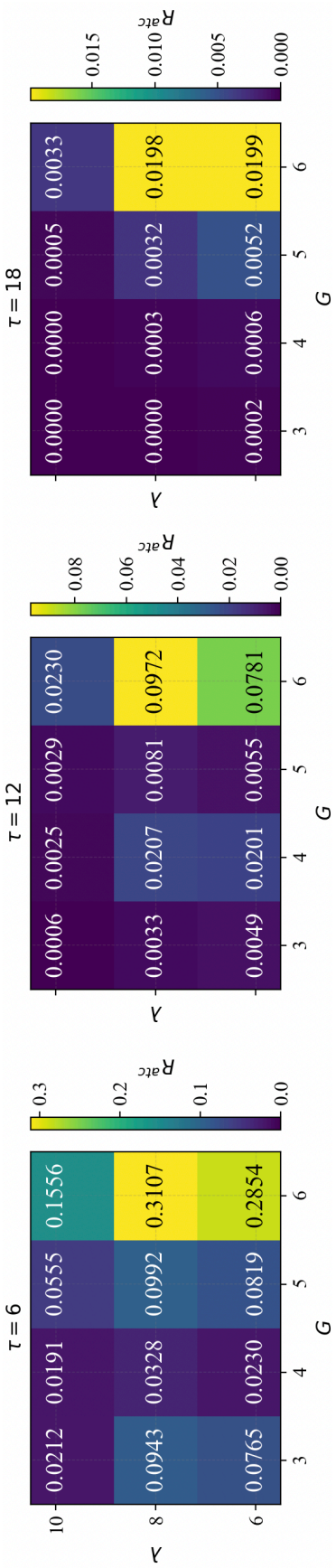

FIGURE 4-4. Evaluation on R^{atc} under different κ, λ, τ and G.

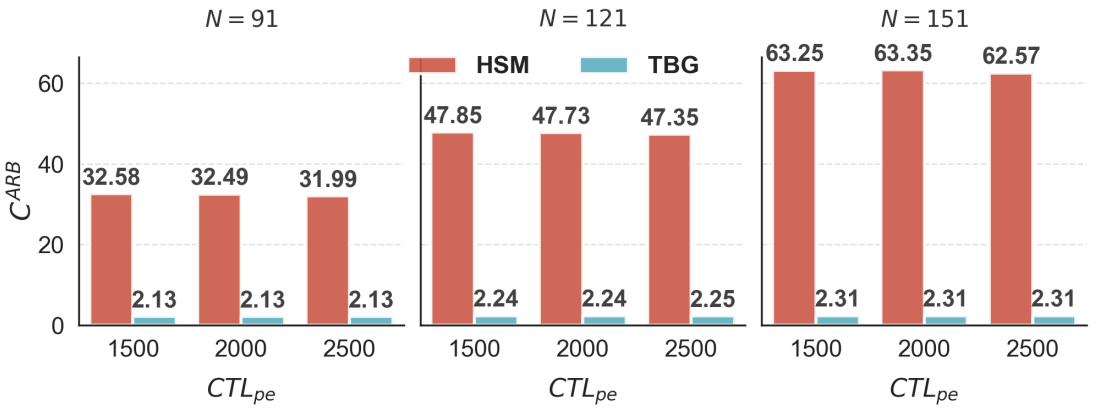

FIGURE 4-5. Comparison of C^{ARB} between HSM and TBG.

4.6.3 Comparison of Redundant Broadcasts between HSM-based P^{ABC} and TBG-based P^{ABC}

This section evaluate the average RB per transaction (C^{ARB}) of HSM-based and TBG-based P^{ABC} under the DFES attack and the random attack strategy of the RSS attack with π^{τ}'s parameter τ=12, the network parameter κ=2, the corruption model's parameter λ=10, ℓ=2, ε=1, p^B=0.1, and the TBG's parameter G=3. By varying replica number N={91, 121, 151} and valid transaction load per epoch CTL^{pe}={1,500, 2,000, 2,500}, Figure 4-5 shows that C^{ARB} under the TBG-based P^{ABC} is no more than 2.31, at least 27.08 times smaller than that of the HSM-based P^{ABC} under the same parameter setting. The low RB of TBG-based P^{ABC} is due to the grouping-strided design which ensures that a BC-delivered transaction is broadcast at most $\max\{g, \tau - g(G-1)\}$ times according to Theorem 4.2. In contrast, although HSM-based P^{ABC} ensures that a valid transaction can only be broadcast exactly once, it remains vulnerable to the DFES attack, which causes each invalid transaction to be broadcast multiple times.

Moreover, as N increases, C^{ARB} of the TBG-based P^{ABC} only shows a slight increase, which is mainly caused by the parameter setting – $(\varepsilon\lambda - \kappa)\ell + p^B N$ Byzantine replicas. Thus, when N increases, the proportion of Byzantine replicas, i.e., $((\varepsilon\lambda - \kappa)\ell + p^B N)/N$, will decrease, making a group have more correct replicas in expectation to broadcast the same transactions. In other words, C^{ARB} of the TBG-

based P^{ABC} is not influenced by *N*. In contrast, C^{ARB} of HSM-based P^{ABC} is significantly influenced by *N*. The reason lies in that under the flooding invalid transaction strategy of the DFES attack, when *N* increases, there are more correct replicas broadcasting the same invalid transactions.

4.6.4 Comparison of Throughput and Latency between HSM-based P^{ABC} and TBG-based P^{ABC} under Different Network Delays

To evaluate the simulated latency, we cover propagation delay P^d (see Eq. (4.9)) ranging from 20ms to 600ms. Figure 4-6 illustrates the comparative performance of the TBG-based P^{ABC} and the HSM-based P^{ABC} under the DFES attack, focusing on average throughput (Tps^A) and latency (L^A).

With the replica number fixed at *N*=91, CTL^{pe}=20,000, and the same parameter setting for other parameters as in Section VII-C, Figure 4-6 shows that the TBG-based framework keeps a pronounced advantage as the propagation delay increases. Specifically, while both protocols experience natural degradation as P^d increases, the gap between them widens in absolute terms. For instance, at $P^d = 600$ ms, TBG reduces the latency L^A by over 86 seconds compared to HSM.

The resilience of TBG-based P^{ABC} in high-latency environments is attributed to the mechanism of π_τ. In HSM-based protocols, the DFES attack causes invalid transactions to accumulate, and high propagation delays exacerbate the issue of accumulating invalid transactions. In contrast, TBG proactively eliminates invalid transactions, making valid transactions timely broadcast and then ABC-delivered.

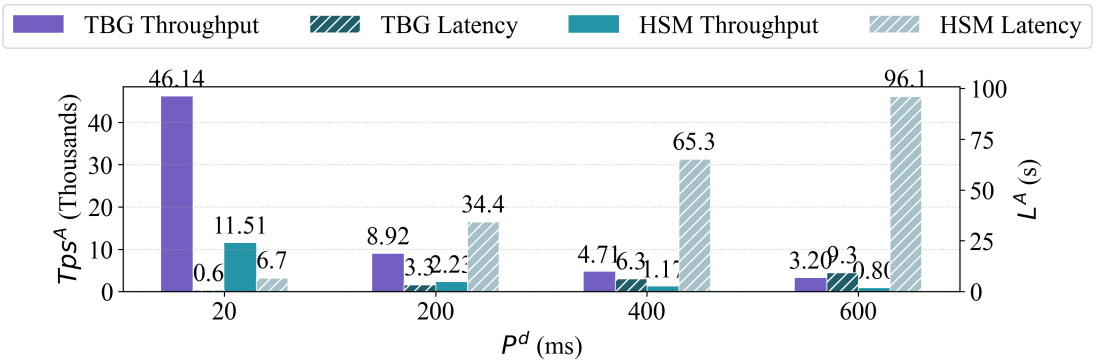

FIGURE 4-6. Evaluation on Tps^A and L^A across different CTL^{pe} and N.

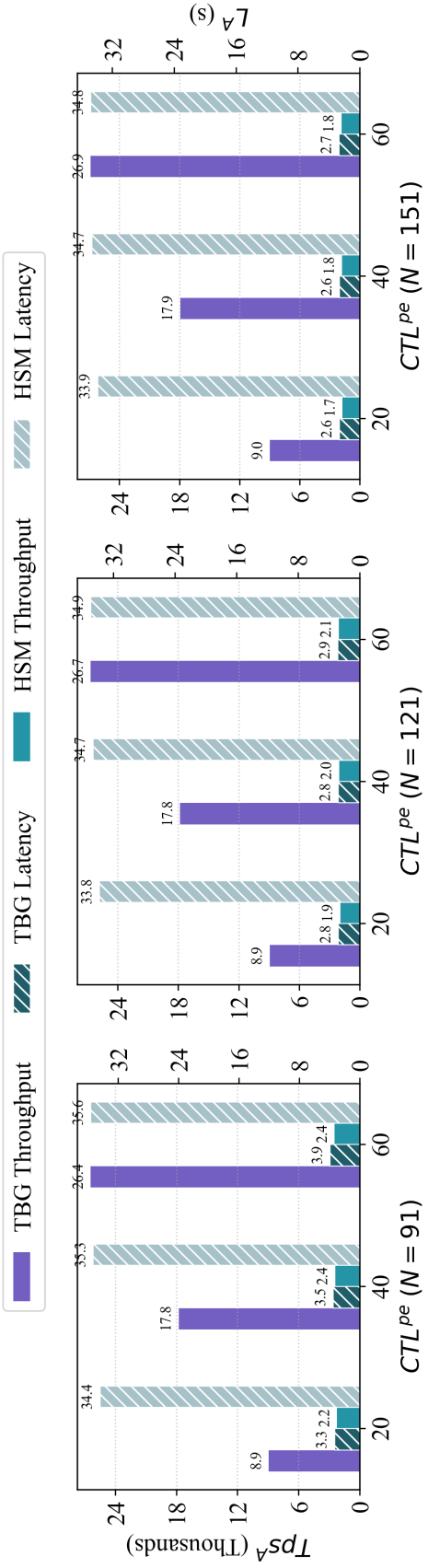

FIGURE 4-7. Evaluation on Tps^A and L^A across different CTL^{pe} and κ.

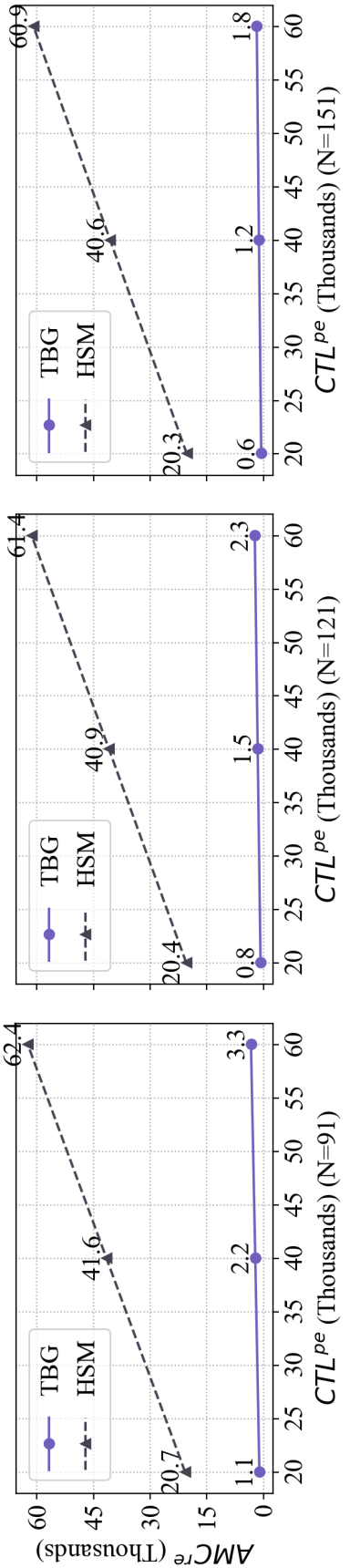

FIGURE 4-8. Evaluation on AMC^{re} across different CTL^{pe}.

4.6.5 Comparison of Throughput and Latency between HSM-based P^{ABC} and TBG-based P^{ABC}

Figure 4-7 illustrates the comparative performance of TBG-based P^{ABC} and HSM-based P^{ABC} under the DFES attack, focusing on throughput (Tps^A) and latency (L^A) across varying replica numbers (N) and valid transaction load per epoch (CTL^{pe}), with the same parameter setting as in Section 4.6.3.

In the simulation, the TBG-based P^{ABC} achieves up to 14.95 times higher throughput and reduces latency by at most a factor of 12.89 compared to the HSM-based P^{ABC}. The results reveal that by deleting invalid transactions and confining each valid transaction to exactly τ replicas, π^{τ} enables valid transactions to be timely batched and broadcast by correct replicas – directly leading to the observed improvements in latency. Moreover, since the average RB per transaction (C^{ARB}) is only 2.2~2.3 under the grouping-strided design of TBG (see Theorem 4.2) according to Figure 4-5, while a group has τ/G=4 replicas under the simulation setting, the results in Figure 4-7 underscore that by reducing RB, the number of unique, ABC-delivered valid transactions significantly increases in each epoch, largely improving the system throughput.

4.6.6 Evaluation on Memory Overhead of Replicas

Finally, we investigate the average memory consumption of a correct replica per epoch AMC^{re} under varying N and CTL^{pe}, as shown in Figure 4-8. The simulation results show that replicas employing the TBG-based P^{ABC} protocol consume up to 33 times less memory compared to those using the HSM-based P^{ABC}. This reduction in memory overhead further supports the effectiveness of the π^{τ} protocol in eliminating invalid transactions and highlights TBG's grouping-strided design in accelerating the broadcasts of valid transactions.

4.6.7 TBG Grouping Latency and Viable Application Scenarios

This section measures the local pre-consensus delay introduced by Transaction Batch Generation (TBG). The measured grouping latency is the time used by a replica to select stored transactions for a batch in Algorithm 4.2. The measurement stops before the replica invokes the batch broadcast primitive. It excludes batch broadcast, block agreement, propagation delay, signature generation and signature verification. The benchmark uses the same implementation environment as the Chapter 4 simulations. The experiment sets B=2,500 and G=3. The received transaction load per replica varies from 1,000 to 120,000 transactions per second. For each load, the grouping operation is repeated 260 times after 30 warm-up runs. Figure 4-9 shows that the median grouping latency remains below 0.027 ms and the 95th percentile remains below 0.046 ms even at 120,000 transactions per second. These values indicate that grouping is a local memory-selection operation. It is performed after storage and before batch broadcast, so it adds no communication round. Its measured cost is small relative to the simulated scoped LBFT latency.

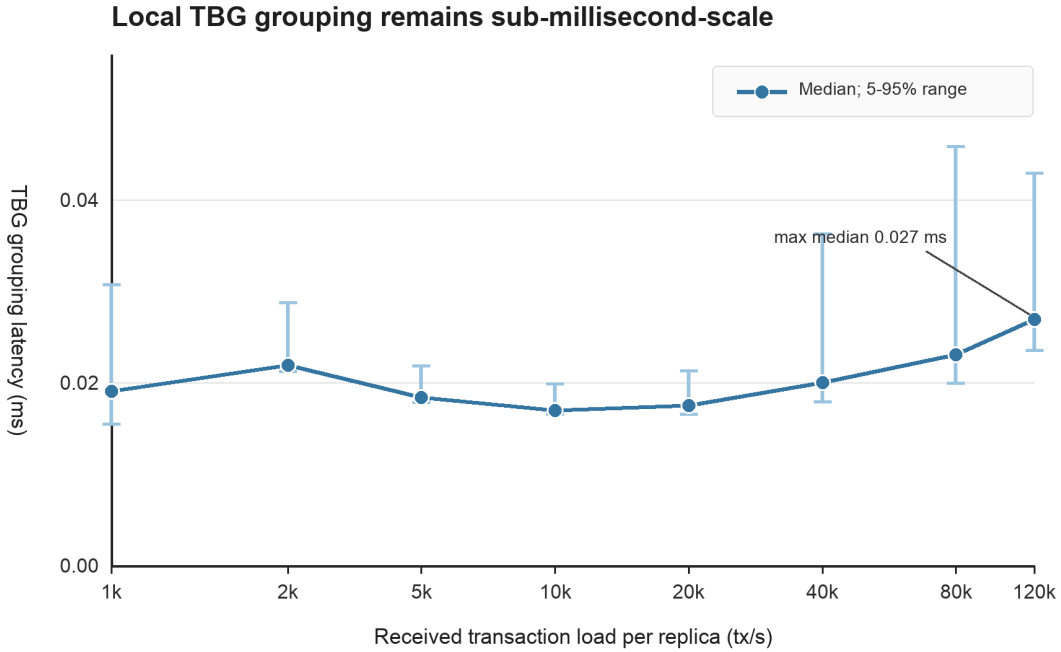

FIGURE 4-9. Local TBG grouping latency under different per-replica received transaction loads.

The measurements also help explain the latency results in Figure 4-6 and Figure 4-7. According to Theorem 4.3, TBG changes only how batch contents are selected. It keeps the batch broadcast primitive and block agreement primitive of scoped LBFT unchanged. The latency benefit therefore comes from reducing invalid and repeated transaction broadcasts before consensus execution, where the transaction management protocol in Section 4.4 deletes invalid transactions and stores each valid transaction only in selected replicas. TBG then assigns each BC-delivered transaction to at most one broadcasting replica group. Correct replicas spend less bandwidth and memory on invalid or repeated transactions, so more valid transactions can be batched in time and then ABC-delivered. Figure 4-6 and Figure 4-7 report this effect as lower simulated latency and higher admitted-transaction throughput under the considered attacks.

TBG-based LBFT is viable when the cyber-layer smart grid task is to admit, order and audit many transactions before an application process uses them. Examples include P2P energy trading orders, settlement metadata before market clearing and electric vehicle (EV) charging-payment transactions. Other examples include demand response commitments, virtual power plant (VPP) event logs and distributed energy resource (DER) status-update transactions [295]. Electricity-carbon-tradable green certificate (TGC) audit transactions also fit this pattern because they require auditable ordering among multiple market actors [296]. Thus, the TBG protocol ensures that honest participants' tranactions (such as trading bids and offers) cannot be censored, but can be quickly processed, and deterministically ordered.

4.7 Chapter Summary

This chapter studies redundant broadcasts (RB) and transaction censorship (TC) in Leaderless Byzantine Fault Tolerant (LBFT) protocols under mobile adversaries. It defines the restricted network model $\mathcal{M}^{AS}$ and the recovery-based corruption model $\mathcal{M}^{RC}$, and analyses how the DFES and RSS attacks affect HSM-based scoped LBFT protocols. Under the tested parameter settings, the TC rate of the HSM-based protocol

reaches 0.76 and the RB value reaches 63. These results identify the security and performance boundary that motivates the proposed transaction-management and batch-generation mechanisms.

To address these limitations, this chapter introduces π^{τ} and TBG as scoped transaction-management and batch-generation mechanisms. The design decouples transaction management and batch generation from the batch broadcast and block agreement primitives. Theorem 4.3 formalises this separation by showing that TBG does not change the two primitives of the scoped LBFT protocol. The simulations show that, under the specified models and attacks, π^{τ} and TBG achieve higher admitted-transaction throughput, lower simulated latency, lower per-replica memory overhead and an approximately zero TC rate in the tested parameter range.

Future work will focus on three directions. First, in partially synchronous environments, the strict epoch constraints, such as the 2κ-strided interval, may be relaxed to reduce latency. Designing an adaptive TBG that estimates and adapts to network conditions is therefore a promising direction. Second, evaluating the TC rate under less restricted asynchronous networks, for example when the skew exceeds the κ setting used in this chapter, is needed to reveal the security boundary of LBFT protocols. Third, examining fairness under the recovery-based corruption model would provide a more complete understanding of LBFT behaviour under mobile adversaries. These directions move beyond the static and adaptive corruption models and can further clarify how LBFT protocols should be deployed in practical environments.

CHAPTER 5

Data Interoperability Protocols for Multi-Blockchains Systems

The proliferation of decentralised applications across different autonomous blockchains raises the need to enable cross-chain data interoperability (CCDI). However, prior approaches for supporting CCDI often hit scalability bottlenecks regarding critical metrics, e.g., memory, or remain prone to withholding and censorship attacks. This Chapter proposes three protocols to implement secure and efficient CCDI under adversarial conditions.

The cross-chain token exchange (CCTE) protocol for atomic swaps is proposed. It adopts a deposit mechanism, a blockchain-of-blockchains (BoB), and Merkle proofs to ensure the completion of token exchanges even under withholding attacks. It utilises a parallelised design to support concurrent token exchanges, thereby improving its efficiency and avoiding censorship attacks that target sequential token exchanges.

The CCDI protocol is proposed to support any CCDI application. It authorises a unique BoB to execute arbitrary CCDI application logic. It integrates a "transfer and in place data update" mechanism to improve its efficiency, and this mechanism enables a blockchain to update its state data items using a single transaction, without requiring any information from other blockchains. Moreover, the CCDI protocol integrates a state data migration scheme, which supports a user to migrate its state data item to censorship-resilient blockchains, and incorporates a malicious user nodes elimination scheme, which enables the updates of state data items in a CCDI process even under withholding attacks.

Systematic performance evaluations are conducted to compare the two protocols with existing ones. The CCTE protocol reduces latency by at least 52% compared to existing protocols under probabilistic consensus setting. The CCDI protocol outperforms prior protocols, lowering communication cost by 59%, computation

overhead by 41%, memory burden by 12%, and latency cost by 33%.

This chapter is based on the author accepted manuscript of: T. Yu, F. Luo, G. Ranzi and J. Wu, "Secure and Efficient Data Interoperability Protocols for Multi-Blockchains Systems," IEEE Transactions on Information Forensics and Security, vol. 20, pp. 11401–11416, 2025, doi: 10.1109/TIFS.2025.3615418.

5.1 Introduction

Since from its proposal in 2008, blockchain (BC) [183] has been extensively practiced in different fields. The wide applications of BC raise the need of Cross-Chain Data Interoperability (CCDI), referring to as a data integration paradigm that the data stored in one BC is reachable, verifiable, and referable by another in a semantically compatible manner [184]. A typical CCDI application is the Cross-Chain Token Exchange (CCTE) (or atomic swaps) [185-192], which allows a party to transfer its tokens to another party in a BC if and only if the latter transfers its tokens to the former in another BC. Besides facilitating CCTE application, CCDI also supports more complicated data interactions across autonomous BCs [193-212] – for example, for social media and Esports applications deployed in different BCs, CCDI can not only enable users in a same BC to interact but also support a user to interact the users from other BCs.

One challenge of CCDI lies in the inherent independence of BCs [183], which restricts that the nodes in a BC can only access the state data (SD) of the BC (e.g., account balances and contracts statuses) and cannot access the SD of other BCs. Such an independence nature of BCs would hinder CCDI operations. Most approaches directly compromise BCs' inde-pendence property to support CCDI. However, lacking BCs' independence may introduce additional issues. For instance, a BC must retain extra information from each connected BC to validate cross-chain data, causing significant memory overhead as the number of connected BCs grows.

Another challenge of CCDI is to guarantee the atomicity property [184] in the presence of withholding and censorship attacks. Specifically, the atomicity property states a situation where all SD items involved in a CCDI process is either fully updated or completely unchanged [184]. Under withholding attacks, a malicious user can reject to submit its transaction, which stalls the updates of all SD items involved in a CCDI process, thereby denying CCDI services [187]. More seriously, under censorship attacks, the atomicity property may not hold anymore when a BC censors a transaction required in a CCDI process, which directly threats CCDI security [183].

5.1.1 State-of-the-Art

A number of protocols have been developed to address the above two challenges in a multi-blockchain system (MBS).

The CCTE scenario. Time-lock-based mechanisms are one of the solutions to enable the independence of BCs and support CCTE. The "lock" operation can be implemented in different ways, such as hash-time locks [187, 188, 190], signature-based locks [189, 192] and time-lock puzzles [191]. The original hash time lock contracts (HTLCs) scheme [187] requires a larger time cost for the "lock" operations to ensure atomicity of a CCTE process, which is time-efficient. Recent work further improves the time efficiency of these mechanisms by moving the locking phase off-chain and employing adaptor signatures to carry out the unlocking phase on-chain [189, 192].

Despite these merits, time-lock-based mechanisms remain vulnerable to withholding and censorship attacks. By design, these mechanisms mandate that one party must complete its token transfer before its counterparty initiates theirs [187-192]. If the party is malicious, it can simply withhold that transfer, stalling the entire conditional exchange and wasting its counterparty's time. Even if the party's token transfer appears on its BC, the counterparty's transfer on the target BC can be censored for a certain time window. During this time window, the party may initiate another

transaction to spend out its to-kens, causing the counterparty's token transfer to fail after the time window. Although fungibility property is a promising solution to counter censorship attacks [189], it breaks down entirely if the party colludes with the target BC.

Recent work [185, 186] proposes a CCTE protocol that preserves the BCs' independence while ensuring resilience to censorship attacks. The protocol in [185] sets up relay nodes to enable CCTE and relies on a blockchain of blockchains (BoB) to ensure the integrity and trustworthy of the relay nodes. [186] extends the CCTE framework to support multi-party scenarios. However, the work in [185] has two significant limitations: firstly, its mechanistic design is not efficient, imposing higher communication, computation, and latency overhead than time-lock-based schemes; secondly, under withholding attacks, the protocol in [185] must roll back all CCTE operations to ensure its atomicity property, thereby denying CCTE services.

Based on the above literature review and discussion, we ask the first research question: *whether there exists a protocol that can (i) ensure the atomicity property under censorship attacks, (ii) progress CCTE services even in the presence of withholding attacks, and (iii) offer substantially higher efficiency, regarding the metrics of communication, computation, memory and latency, than the CCTE protocol in* [185]?

The CCDI scenario. There are two paradigms of CCDI protocols to support complex CCDI applications, e.g., social media and Esports. In the first paradigm, a BoB is introduced in an MBS, but its role is simplified to just forward or trans-late a CCDI request from the source BC to the target BC [193-202]. On this basis, the CCDI protocols facilitate direct state data verification between BCs, which thus require each BC to store the block header of the others – this in turn compromises the independence property of involved BCs.

The second paradigm of CCDI protocols also relies on a BoB, but it assigns the BoB with enhanced abilities [203-212]. Specifically, the BoB in [206-208] can create SD items from its connected BCs and execute any CCDI logic. The BoB can also issue a statement (also known as a certificate [206-209] or a proof [210-212]) for a CCDI

request and send it to relevant BCs. However, to verify the BoB's statements, the BC needs to cache the extra information (e.g., block hashes of the BoB), which compromises the independence property of these BCs, thereby burdening them with extra memory cost.

Notably, the protocols of the two paradigms [193-212] execute the operations for the updates of SD items within a single BC – either a BoB or a target BC. With this design, the protocols can be censorship-resilient through selecting a censorship-resilient BC as the BoB or the target BC. However, these protocols still have two disadvantages. First, when a CCDI process operates under these protocols, it is still vulnerable to withholding attacks. This is because these protocols can only roll back the whole operations of the CCDI process when withholding attacks occurs, rather than further improving the quality of the atomicity property to prevent from denying CCDI services. Second, they hardly preserve the independence property of involved BCs, as these BCs need to store information to validate other BCs' proofs or BoBs' statements.

Based on the above analysis, we ask the second research question: whether there exists a protocol that can (i) progress CCDI services even in the presence of withholding attacks, (ii) preserve BCs' independence, and (iii) offer higher efficiency, regarding the metrics of communication, computation, memory and latency, than the state-of-the-art CCDI protocols?

5.1.2 Contributions of This Chapter

This study is devoted to providing solutions to the aforementioned two open research questions. The overarching contribution of this paper is to propose new protocols for enabling more secure and efficient data interoperability among multiple BCs. The contributions of this paper are 4-fold:

1. To propose a new CCTE protocol. Compared with the CCTE protocol in [185], the proposed protocol has 4 advantageous features: First, it does not need relay nodes. As a result, the protocol not only leads to a less memory cost compared with the

protocol in [185] but also makes the CCTE process only involve two token transfer transactions, one less than the protocol in [185]. Second, the two token transfer transactions are executed in parallel, offering greater efficiency compared with the protocol in [185] that requires sequential execution of the three token transfer transactions. Third, unlike the protocol in [185] which assumes a 1:1 exchange rate for different token types, or the protocols [187-192] that assume pre-decided exchange rates, the proposed protocol is based on an Automatic Market Maker mechanism [214] and the cross-chain token claim scheme [193] with some modifications – with this design, the protocol can be used to provide practical exchange rates without compromising the BCs' independence. Last, it can complete all token transfer transactions even in the presence of censorship and withholding attacks; in contrast, the protocol in [185] needs to roll back all CCTE operations to preserve its atomicity property, leading to the denial of CCDI services.

2. To propose a new CCDI protocol. It only costs three transaction execution phases to complete a CCDI process, and our solution is optimal compared with the current protocols [193-212]. Specifically, after locking all SD items involved in a CCDI process (Phase 1), their metadata is submitted to a BoB, and the BoB then executes the CCDI operations to update these metadata (Phase 2). The updated metadata is finally sent back to its BCs (Phase 3). Since storing and updating metadata is more computationally efficient than creating a SD item in a BC, this design is more lightweight than the widely adopted "mint or claim"-based mechanisms [215, 216].

Furthermore, we propose a threshold-signature-based "trans-fer and in place data update" mechanism and integrate it into the CCDI protocol. The mechanism preserves each participating BC's independence and offers more efficient performance than the "lock and claim" mechanisms [215, 216]. Specifically, all BoB nodes collectively create an account represented by a threshold public key, and the account is registered on each participating BC. At the start of a CCDI process, the relevant SD items in a BC are transferred to the account for locking operation. Once the BoB yields updated metadata, the updated metadata plus a threshold signature [217] are submitted back to

the BC. The BC verifies the signature and updates the locked SD items according to the updated metadata. Since the thresh-old public key of the account can verify the signature, no external data (e.g., BoB's or other BCs' root hashes) are required, thereby retaining each BC's independence. Moreover, a trans-action containing a single threshold signature can simultaneously update multiple SD items in a BC, which further boosts the efficiency of the CCDI protocol.

As another critical part of the proposed protocol, a state-data migration scheme and a malicious user-node elimination scheme are proposed and integrated into the CCDI protocol. The former scheme enables a user to migrate its SD item from a censorship-prone BC to a censorship-resilient BC without compromising either BC's independence. The latter scheme identifies the users that launch withholding attacks. By eliminating the identified users, SD items of honest users can be further updated, instead of rolling back all operations involved in a CCDI process to preserve atomicity property.

3. To establish an analysis framework to prove the proposed CCTE and CCDI protocols satisfy the 3 properties that are outlined in [213] as necessary properties of a cross-chain protocol, i.e., atomicity, isolation and durability . Besides these three properties, we also prove the proposed CCTE protocol is with the fungibility and script-minimum properties . The fungibility property is essential for mitigating censorship attacks, while the script-minimum property enhances the compatibility of a CCTE protocol by enabling BCs that do not support smart contracts to participate in the CCTE process.

4. To implement the proposed protocols on Ganache [218] and Remix-IDE Ethereum [219]. Comprehensive case studies are conducted to evaluate the protocols by comparing with the distinguished protocols [185, 192, 208, 210]. The evaluation results demonstrate that the proposed protocols are with lower communication, computation, memory and latency cost.

The rest of this Chapter is organised as follows. Section 5.2 introduces some basic concepts related to this research. Section 5.3 provides an overview of the proposed protocols. Technical details of the proposed CCTE and CCDI protocols are presented

in Sections 5.4 and 5.5, respectively. Simulation is re-ported in Section 5.6. Section 5.7 concludes the paper and discusses possible future work.

5.2 Basic Concepts

This section introduces some basic concepts that are preliminaries of understanding the proposed protocols.

5.2.1 Blockchain

A BC is a distributed ledger comprising a sequence of blocks, where each block cryptographically links to its predecessor. A BC could be associated with the independence property defined below.

Independence property: For a BC with a set of verification nodes $\mathcal{V}$. The independence property of the BC holds if every node $v \in \mathcal{V}$ only stores the state data of the BC and does not store the state data of any other BCs.

5.2.2 Consensus Protocols

A consensus protocol [220] refers to as a set of instructions executed by a BC's nodes to achieve agreement on a block of transactions. A secure consensus protocol is considered to have safeness and liveness properties at the same time [183]. The safeness property ensures all transactions are correctly executed and are executed in a globally consistent order. The liveness property ensures that all the valid transactions submitted to a BC will eventually be stored into the BC.

The consensus protocols can be generally divided into deterministic protocols and probabilistic protocols [221]. With deterministic consensus protocols, once a block is generated, the transactions within it become immutable. In contrast, ac-cording to the common prefix property [222], probabilistic protocols ensure that once a block is extended by a sufficient number of subsequent blocks, transactions included in this block becomes immutable with high probability.

5.2.3 Smart Contracts

A smart contract is a code container deployed in BCs, which contains deterministic algorithms [223]. The execution of a smart contract can be represented as:

$$S_{sc}^{new} \leftarrow SC(S_{sc}, Txs(SD, params)) \tag{5.1}$$

where $SC(\cdot)$ denotes the smart contract; S_{sc} is the current state information of $SC(\cdot)$; $Txs(SD, params)$ represents the transactions which invoke $SC(\cdot)$, and each transaction contains state data SD and necessary parameters (denoted as $params$), e.g., signatures; S_{sc}^{new} is the output generated by the smart contract, e.g., the updated state of $SC(\cdot)$. The equation of Eq. (5.1) represents a state transition of $SC(\cdot)$ from S_{sc} to S_{sc}^{new}.

5.2.4 Cross-Chain Data Interoperability Protocol

A CCDI protocol defines a set of executable instructions. Execution of a CCDI protocol will lead to a state transition of all the BCs involved in a CCDI process. A CCDI process defined by a CCDI protocol includes the following elements:

(i) A set of BCs $\boldsymbol{\mathcal{B}} = \{BC_1, BC_2, ...\}$ and a group of users $\boldsymbol{\mathcal{U}} = \{U_1, U_2, ...\}$. Both are considered as participants of the CCDI process.

(ii) A list of transactions to be executed denoted as $\boldsymbol{\mathcal{T}} = \{Tx_1, Tx_2, ...\}$. Each transaction in $\boldsymbol{\mathcal{T}}$ is generated by a specific user from $\boldsymbol{\mathcal{U}}$ and it will be executed in a specific BC from $\boldsymbol{\mathcal{B}}$.

(iii) Execution phases of the CCDI process, denoted as $\boldsymbol{\mathcal{P}} = \{P_1, P_2, ...\}$. The phases are executed in a sequential manner, and each phase involves the execution of one or more transactions. To execute a phase, all the transactions involved can be concurrently executed.

5.2.5 Merkle Proofs

A Merkle proof [198] certifies the existence of a transaction in a block. Key operations for Merkle proofs include:

(i) $Gen_{proof}(Tx,\ BH)$. It means the creation of a Merkle proof for a transaction Tx in a block with its block header BH.

(ii) $Val_{proof}(P,\ Tx,\ R_{BH})$. It represents the verification of a Merkle proof P. R_{BH} is a root hash of a block header BH. The verification operation returns true if Tx is in the block with a header of BH; it returns false otherwise.

5.2.6 Threshold Signature

A threshold signature (TS) [217, 293] provides evidence, under a threshold public key, that at least a threshold number of nodes have signed a message *m*. Consider a BC with $M = 3f + 1$ consensus nodes, where f is the maximum number of Byzantine consensus nodes. If *m* is valid, each participating node signs *m* to generate a TS share. A valid TS can then be constructed from at least $2f + 1$ distinct TS shares and verified using the threshold public key. In the CCDI protocol, the TS is used as compact evidence that the updated SD items have been validated by the required consensus threshold, so the smart contract in each participating BC can verify the evidence locally. Recent Boneh-Lynn-Shacham (BLS) threshold-signature implementations on the BLS12-381 curve report millisecond-level share-generation and share-verification costs. Das and Ren [293] report 0.81-3.92 ms to generate one TS share and 0.76-2.16 ms to verify one TS share. They also report 55.43-149.52 ms to construct one TS from 64 TS shares, and state that final TS verification is identical to standard BLS verification. In this chapter, the latency comparison is measured at the level of transaction-execution phases and block-generation intervals. These reported TS costs are therefore computationally acceptable for the CCDI workflow considered here and are not expected to dominate the reported latency comparison (see Section 5.6).

5.3 System and Threat Models, Assumptions and Goals

This section provides an overview of the system to which the proposed protocols apply, together with the threat models, assumptions and design goals of the protocols.

5.3.1 System Model

Our system model is based on a blockchain and blockchains-based multi-blockchains system (BoB-MBS). A BoB-MBS consists of a BC acting as a BoB and multiple BCs (called the external BCs). We denote the number of external BCs in a BoB-MBS as N and denote the external BCs as BC_i, $i \in [N]$, where $[N]$ is short for the set $\{1,2,\dots,N\}$. For BC_i or the BoB, it progresses in consecutive epochs, with the e_i-th or e_B-th epoch indexed by e_i or e_B ($e_i \in \mathbb{N}, e_B \in \mathbb{N}$). At the end of each epoch, a block is generated in a BC. All BCs in the system operate under the semi-synchronous network model [220] and the adaptive corruption model [221]. In semi-synchronous networks, there exists an unknown time duration Δ on message delivery delays, so a block generation time is not fixed. In the adaptive corruption model, an adversary may, at any time point during the execution of a consensus protocol, corrupt up to f consensus nodes in a BC.

5.3.2 Threat Model

Two attack scenarios (i.e., censorship attacks and withholding attacks) are considered in the BoB-MBS system, aiming to deny CCDI services or make a CCDI protocol violate its atomicity property.

Censorship attacks [226]. In a normal case, a transaction Tx sent to a blockchain BC_i ($i \in [N]$) in epoch e_i is immutably stored in BC_i in epoch $e_i + \kappa_i + 1$, where $\kappa_i \geq 0$ is a parameter of the common prefix property [222] of BC_i. A censorship attack delays the immutable inclusion of Tx in BC_i to epoch $e_i + \kappa_i + 1 + D_i$, where D_i is a finite positive integer.

Withholding attacks. In a CCDI protocol, a normal case is that the user node participating in a CCDI process should send its transaction Tx to a specified BC (see Section 5.2.4) before a deadline defined by a CCDI protocol. A withhold attack occurs when the user node either submits Tx after this deadline or fails to submit it throughout the CCDI process.

5.3.3 Protocols Assumption and Design Goals

Assumptions. The proposed CCDI protocols work in the BoB-MBS. The following assumptions are applied to ensure the security of the protocols:

(i) A deterministic consensus protocol is used in the BoB. (i) A deterministic consensus protocol is used in the BoB.

(ii) The consensus nodes in the BoB store both the BoB's state data and the latest κ_i block headers of BC_i ($i \in [N]$).

(iii) Every BC and the BoB in the BoB-MBS hold the safeness and liveness properties.

(iv) The BoB and some BCs in the BoB-MBS are censorship-resilient due to their protocols' design [223-225].

Goals. It is expected that the proposed protocols can achieve the following design goals:

The independence property of each external BC in the BoB-MBS is preserved when these BCs involve in the CCDI processes defined by the proposed protocols.

The protocols hold the atomicity, isolation, and durability properties. These properties are defined as follows:

Atomicity: *In a CCDI process, the states of all the participants either successfully transit to new states or remain unchanged.* Note that the atomicity property does not apply to the changes of a participant's state that is caused by the operations unrelated to the CCDI process.

Isolation: *In a phase of the CCDI process when multiple transactions need to be concurrently executed, each transaction must be executed independently.*

Durability: *When a transaction Tx_1 must be executed in prior to another transaction Tx_2, the change of the BoB-MBS's state caused by Tx_1 must be permanently and immutably stored in the corresponding BC before executing Tx_2.*

Besides the above properties, the proposed CCTE protocol (see Section 4.4) holds two additional properties:

Fungibility: *For observers except for transaction formation agents, token receivers and the BoB, they cannot distinguish a token transfer transaction in a CCTE and a standard token transfer transaction.*

Script-minimum: *An external BC only needs signature verification scripts to participate in a CCTE process.*

5.4 Cross-Chain Token Exchange Protocol

The proposed CCTE protocol (denoted as Π_{CCTE}) is designed for CCTE (atomic swaps) scenarios where a sender intends to use its tokens of type $Token_i$ to exchange for a receiver's x_j tokens of type $Token_j$, and these two types of tokens circulate on BC_i and BC_j, respectively. The CCTE process defined in Π_{CCTE} involves 5 basic transactions and 5 participants. The transactions include a cross-chain transaction (CCT), two native transactions (denoted as NT_s and NT_r), and two proof transactions (denoted as PT_s and PT_r). The participants include a sender, a receiver, the two blockchains BC_i and BC_j, and the BoB. In some situations, an additional cross-chain transaction CCT_V and at most two fraud-proof transactions (each denoted as T_{FP}) (see Phase 3 of this section) will be needed. The type of the token circulating in the BoB is denoted as $Token_B$.

As illustrated in Figure 5-1, the CCTE process comprises of 3 phases: firstly, the sender sends the CCT to the BoB for activating the CCTE process. Secondly, the sender sends the native transaction NT_s to BC_i and the BoB for transferring its tokens with type $Token_i$ to the receiver's address in BC_i; in the same time, the receiver sends the native transaction NT_r to BC_j and the BoB for transferring its tokens with type $Token_j$ to the sender's address in BC_j. Phase 3 will be executed for security purposes, in which the proof transactions PT_s and PT_r will be sent to the BoB. The transaction execution logics are supported by four smart contracts deployed in the BoB. Table 5-1 provides a brief description on their functionalities, while their roles will be explained later in this section.

TABLE 5-1. Smart contracts used in Π_{CCTE}

Notation	Functionality
SC_D	Lock $Token_B$
SC_T	Execute cross-chain transactions
SC_{AMM}	Calculate exchange rates
SC_{MPV}	Validate received Merkle proofs

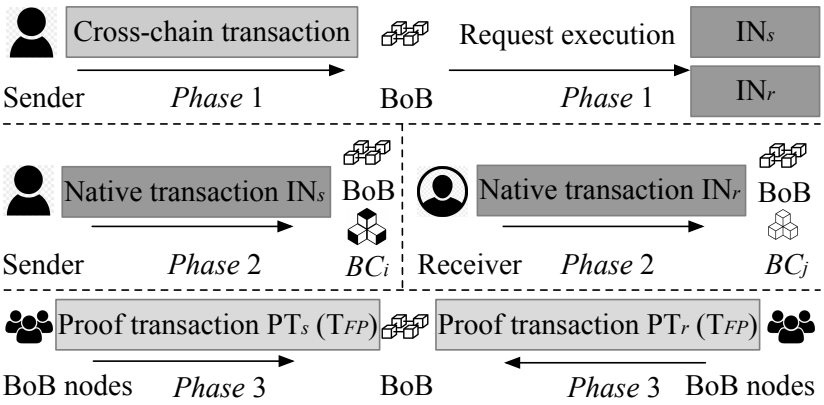

FIGURE 5-1. The transactions of Π_{CCTE} in each phase.

5.4.1 Workflow of Π_{CCTE}

The overall procedure of CCTE is shown in Algorithm 5.1. To defend against censorship and withholding attacks, before the CCTE process, the sender and the receiver need to deposit a certain amount of $Token_B$ (denoted the amounts as $x_{s,B}$ and $x_{r,B}$, respectively) in the BoB through the smart contract SC_D (ln 01). The CCTE process is performed with the three phases:

Phase 1: The sender generates a cross-chain transaction CCT and sends it to the BoB to invoke the smart contract SC_T (ln 02). SC_T requests two exchange rates $E_{i,j}$ and $E_{B,j}$ from the smart contract SC_{AMM} (ln 03), where $E_{i,j}$ is the exchange rate between $Token_i$ and $Token_j$ and $E_{B,j}$ is the exchange rate between $Token_B$ and $Token_j$. These two rates are determined following an exchange rate determination scheme, which will be elaborated in Section 5.4.2. Based on the exchange rates, SC_T calculates the amount of $Token_i$ or $Token_B$ needed for exchanging x_j $Token_j$ (ln 04).

To validate the cross-chain transaction CCT, the smart contract SC_T checks whether the sender and the receiver have enough deposits of $Token_B$ by invoking the smart

contract SC_D (ln 05). If the sender or the receiver does not have enough deposits, the CCT is discarded, which can prevent participators from launching withholding attacks afterwards (ln 09). If they have, SC_T generates two instructions IN_s and IN_r (lns 06 and 07): IN_s is for the sender to transfer x_i $Token_i$ to the receiver in BC_i; IN_r is for the receiver to transfer x_j $Token_j$ to the sender in BC_j. Aiming at quickly terminating the CCTE process, SC_T sets a deadline before which proof transactions on IN_s and IN_r must be included to the BoB (ln 08).

Phase 2. The sender and the receiver generate native transactions $NT_s(IN_s)$ and $NT_r(IN_r)$ for the two instructions IN_s and IN_r, respectively. They send $NT_s(IN_s)$ and $NT_r(IN_r)$ to BC_i and BC_j, respectively. In the meantime, they send $NT_s(IN_s)$ and $NT_r(IN_r)$ to the BoB for censorship and withholding resistance purposes (lns 10 and 11), as the two sending operations enable the consensus nodes in the BoB to submit $NT_s(IN_s)$ and $NT_r(IN_r)$ to corresponding BCs. If the sender and the receiver are honest and no attack exists, the CCTE process terminates at the end of Phase 2; otherwise, Phase 3 is needed:

Phase 3. If the native transactions $NT_s(IN_s)$ and $NT_r(IN_r)$ are stored in the corresponding BCs, the sender and the receiver generate Merkle proofs P_s and P_r for $NT_s(IN_s)$ and $NT_r(IN_r)$, respectively (lns 12 and 13). Each Merkle proof and the corresponding native transaction are packaged into a proof transaction. The proof transactions are sent to the BoB to invoke the smart contract SC_{MPV} (lns 14 and 15). SC_{MPV} validates the received proofs and records the valid Merkle proofs (lns 16-18). Then, there are two situations to confirm the CCT:

Situation 1: If SC_{MPV} receives the two valid proofs, the CCTE process completes (lns 19 and 20).

Algorithm 5.1 CCTE Procedures

Notations

$*= i$ or j; e_*: BC_*'s epoch index in which CCT is sent to the BoB; $PK_{s,i}$, $PK_{s,j}$ and $PK_{s,B}$: IDs of the senders in BC_i, BC_j and the BoB, respectively; $PK_{r,i}$, $PK_{r,j}$ and $PK_{r,B}$: IDs of the receiver in BC_i, BC_j and the BoB, respectively; $\bar{\kappa}$: the secure parameter ($\bar{\kappa} \geq 2$); $Max()$: a function returning the largest value among the inputted values; $Min()$: a function returning the smallest value among the inputted values; $BH_*^{e_*'}$: the header of BC_*'s block generated in epoch e_*'; $R(BH_*^{e_*'})$: the root hash of $BH_*^{e_*'}$. FP: the fraud-proof generated by consensus nodes of the BoB. e_*^c: the current epoch index of BC_*. $x_{s,i}$: the number of $Token_i$ in the sender's account in BC_i; $x_{s,j}$: the number of $Token_j$ in the receiver's account in BC_j.

CCTE instance: exchange for x_j $Token_j$ using tokens with type $Token_i$.

01 SC_D: $PK_{s,B}$ =[$Token_B$: $x_{s,B}$]; $PK_{r,B}$ =[$Token_B$: $x_{r,B}$]

Phase 1 – One cross-chain transaction execution

02 $PK_{s,B} \rightarrow CCT(BC_i, BC_j, PK_{s,i}, PK_{s,j}, PK_{r,i}, PK_{r,j}, PK_{r,B}, x_j, Token_i, Token_j, SC_T) \rightarrow$ BoB

03 $E_{i,j} = C_{AMM}(Token_i, Token_j)$; $E_{B,j} = SC_{AMM}(Token_B, Token_j)$

04 $x_i = x_j / E_{i,j}$; $x_B = x_j / E_{B,j}$

05 $SC_T \rightarrow SC_D$: if $x_{s,B} > \bar{\kappa} \times x_B$ and $x_{r,B} > \bar{\kappa} \times x_B$:

06 SC_T: IN_s =(From: $PK_{s,i}$, To: $PK_{r,i}$, Transfer: x_i $Token_i$)

07 IN_r =(From: $PK_{r,j}$, To: $PK_{s,j}$, Transfer: x_j $Token_j$)

08 $Max(e_i + \bar{\kappa} + \kappa_i, e_j + \bar{\kappa} + \kappa_j)$; $e_* + \bar{\kappa}$: the epoch index of BC_* when CCT is stored

09 Else: discard CCT of ln 02 and terminate

Phase 2 – Two native transactions execution concurrently

10 Sender: $PK_{s,i} \rightarrow NT_s(IN_s) \rightarrow BC_i$ and BoB

11 Receiver: $PK_{r,j} \rightarrow NT_r(IN_r) \rightarrow BC_j$ and BoB

Phase 3 – Two proof transactions execution concurrently

12 $P_s = Gen_{proof}(NT_s(IN_s), BH_i^{e_i'})$

13 $P_r = Gen_{proof}(NT_r(IN_r), BH_j^{e_j'})$

14 $PT_s(P_s, NT_s(IN_s), R(BH_i^{e_i'}), SC_{MPV}) \rightarrow$ BoB

15 $PT_r(P_r, NT_r(IN_r), R(BH_j^{e_j}), SC_{MPV}) \rightarrow$ BoB

16 SC_{MPV}: (i) $CCT \in SC_T$;(ii) $e_*' > e_* + \bar{\kappa}$; (iii) $Val_{proof}(P, NT, R(BH_*^{e_*'}))$

	If true is returned for all the three checking operations:
17	$P \in SC_{MPV}$
18	Else: discard P
19	Situation 1: If $(P_s, P_r) \in SC_{MPV}$
20	SC_{MPV}: Confirm CCT of ln 02
21	Situation 2: If P_s and/or $P_r \notin SC_{MPV}$ and $Min(e_i^c, e_j^c) \geq Max(e_i + \bar{\kappa} + \kappa_i, e_j + \bar{\kappa} + \kappa_j)$
22	The nodes of the BoB: $CCT_V(SC_{MPV}) \rightarrow$ BoB
23	SC_{MPV}: execute ln 21; if true is returned, lns 24-29; else: ln 30
24	Sub-situation 1: If $NT_s(IN_s) \notin BoB$: SC_{MPV}: (From: $PK_{s,B}$, To: $PK_{r,B}$, Transfer: $x_B\ Token_B$), confirm CCT of ln 02
25	Sub-situation 1: If $NT_r(IN_r) \notin BoB$: SC_{MPV}: (From: $PK_{r,B}$, To: $PK_{s,B}$, Transfer: $x_B\ Token_B$), confirm CCT of ln 02
26	Sub-situation 2: If $NT_s(IN_s) \in BoB$: execute ln 14 and invoke ln 19
27	Sub-situation 2: If $NT_r(IN_r) \in BoB$: execute ln 15 and invoke ln 19
28	Sub-situation 3: If $x_{s,i} < x_i$: generate T_{FP} to invoke lns 16 and 24
29	Sub-situation 3: If $x_{r,i} < x_j$: generate T_{FP} to invoke lns 16 and 25
30	SC_{MPV}: Discard $CCT_V(SC_{MPV})$

*Note: For simplicity, the signatures contained in the transactions are omitted. This applies to the rest of this paper

Situation 2: If one or both valid proofs fail to be recorded in SC_{MPV} by the deadline defined in ln 08, a transaction CCT_V is then generated by a node of the BoB to invoke SC_{MPV} (lns 21 and 22). SC_{MPV} then checks whether the situation defined in ln 21 is true (ln 24). If it is false, CCT_V is discarded (ln 30). Otherwise, the following 3 sub-situations may happen:

Sub-situation 1: If either $NT_s(IN_s)$ or $NT_r(IN_r)$ fails to appear in the BoB by its deadline (ln 08) due to withholding attacks, SC_{MPV} transfers the sender's (or the receiver's) deposited tokens with type of $Token_B$ to the counterparty's account in the BoB and the CCTE process completes (ln 24 or ln 25).

Sub-situation 2: If $NT_s(IN_s)$ (or $NT_r(IN_r)$) is recorded in the BoB, $f+1$ pre-defined consensus nodes of the BoB obtain $NT_s(IN_s)$ (or $NT_r(IN_r)$) and sends it to BC_i (or BC_j). If $NT_s(IN_s)$ (or $NT_r(IN_r)$) is stored in BC_i (or BC_j), the $f+1$ consensus nodes generate its proof transaction (ln 14 or ln 15) and send it to the BoB for further execution (ln 26 or ln 27).

Sub-situation 3: Before $NT_s(IN_s)$ (or $NT_r(IN_r)$) is stored in BC_i (or BC_j), if any of the $f+1$ nodes finds that the balance of the sender's $Token_i$ (or the receiver's $Token_j$) on BC_i (or BC_j) falls below x_i (or x_j), it generates a fraud-proof transaction T_{FP} with a Merkle proof and send T_{FP} to the BoB (ln 28 or ln 29).

5.4.2 Token Exchange Rate Determination Scheme

We design a Token exchange rate determination scheme to enable the smart contract SC_{AMM} to promptly determine the exchange rates between two types of tokens circulating in the BoB-MBS. The scheme comprises two components: an Automated Market Maker (AMM) and a Cross-chain Token Claim Protocol (Π_{CCTC}). We will present the two components first, followed by the procedure of the token exchange rate determination scheme.

Automated Market Maker. AMM uses a Liquidity Pool (LP) and mathematical

algorithms to calculate exchange rate(s) between two types of tokens in a BC. The LP is deployed in the BoB, and it can be considered as a marketplace for token providers to save different types of tokens. For illustration, AMM considers the Balancer protocol [214] to calculate the exchange rate. For simplicity, we assume that there are N types of tokens in the system, where each BC has one type of token. During the initiation of the LP, the AMM assigns a pre-defined weight to each token type (denoted as $w_i,\ i \in [N]$) with a condition of $\sum_{i=1}^{N} w_i = 1$. It then determines the exchange rate between two different types of tokens as:

$$E_{i,j} = \frac{B_j}{B_i} \times \frac{w_i}{w_j} \tag{5.2}$$

where $E_{i,j}$ is the exchange rate between $Token_i$ and $Token_j$; B_i and B_j are the quantity of the two types of tokens in the LP. B_i and B_j will be quantified in the LP based on the cross-chain token claim protocol (denoted as Π_{CCTC}):

Cross-chain token claim protocol. Based on the token exchange rates generated from the AMM, Π_{CCTC} is designed to facilitate a token provider to save a specific type of token from the corresponding external BC to the LP. Π_{CCTC} is based on the cross-chain token claim scheme in [193] with modifications. Considering a token provider intends to save x tokens with the type of $Token_*$ to the LP, the protocol works as follows:

(i) The BoB validating nodes jointly create a threshold public key-based address [217] in each external BC BC_i. Denote the address as $TPK\_Addr_i,\ i \in [N]$.

(ii) The token provider sends the x tokens with the type of $Token_*$ to an address TPK_Addr_* for locking operation.

(iii) The token provider generates a Merkle proof for the token transfer operation in (ii). Based on the proof, SC_M mints x $token_*$ in the BoB and transfers the minted tokens to the token provider's address in the BoB.

(iv) The token provider transfers the x $Token_*$ from its address in the BoB to the AMM smart contract SC_{AMM}.

Procedure of token exchange rate determination. With Π_{CCTC}, any types of

tokens from an external BC can circulate in the BoB. The token exchange rate is determined via 3 steps:

(i) Smart contract deployment. The AMM is implemented in SC_{AMM} that is deployed in the BoB. There are two data structures in the LP of SC_{AMM}: $\mathbf{LP}_V$ and $\mathbf{LP}_{KV}$. $\mathbf{LP}_V$ stores N pre-defined weighting values, one for a type of token. $\mathbf{LP}_{KV}$ stores N key-value pairs. For each pair, the key is the ID of a specific token type, and the value is the number of tokens with this type.

(ii) Initialise the LP. SC_{AMM} invokes a function (denoted as $\mathcal{F}_{VI}$) to initialise the N weights in $\mathbf{LP}_V$:

$$(w_1, \dots, w_N) = \mathcal{F}_{VI}(Token_1, \dots, Token_N) \tag{5.3}$$

(iii) Determination of the token exchange rate. Token providers contribute their claimed tokens to SC_{AMM}. SC_{AMM} stores the key-value pairs of the tokens into $\mathbf{LP}_{KV}$. Once $\mathbf{LP}_V$ and $\mathbf{LP}_{KV}$ are constructed, Eq. (5.2) can be applied to calculate the exchange rate between any two types of tokens.

5.4.3 Security Analysis

This section is devoted to analysing the security of the proposed CCTE protocol. The analysis is represented by the following theorems:

Theorem 5.1. *For the CCTE process defined in Π_{CCTE}, the independence and the script-minimum properties of each participating BC_i ($i \in [N]$) in the BoB-MBS can be preserved.*

Proof. Consider a participating BC_i ($i \in [N]$). In the second phase of the CCTE process, a token transfer transaction NT_s will be executed in BC_i. To execute NT_s, BC_i only needs to check whether the sender of NT_s has sufficient tokens, which does not need the external state data. Thus, the independence property is held by BC_i. Since the execution of the transaction NT_s only requires BC_i to have the scripts for validating the signature in NT_s, the script-minimum property is also preserved. □

Theorem 5.2. *Under the corruption and network models defined in Section 5.3.1,*

if every participating BC_i *(*$i \in [N]$*) and the BoB holds the safeness and liveness properties,* Π_{CCTE} *will preserve the atomicity, isolation, durability properties.*

Proof. For the atomicity property: the CCTE process has two state transitions: (a) the sender consumes x_i $Token_i$ (or x_B $Token_B$) and obtains x_j $Token_j$ (or x_B $Token_B$). And (b) the receiver obtains x_i $Token_i$ (or x_B $Token_B$) and consumes x_j $Token_j$ (or x_B $Token_B$). There are three situations:

(i) Situation 1 – the CCT is invalid. In this situation, the CCT will be discarded and the CCTE process will terminate without any state transition (see lns 05 and 09 of Algorithm 5.1).

(ii) Situation 2 – the CCT is valid, and both the sender and the receiver are honest. Once the CCT is validated to be valid, two instructions (i.e., IN_s and IN_r in Algorithm 5.1) will be generated. Also, the honest sender and the honest receiver will send NT_s and NT_r to BC_i and BC_j, respectively. In the meantime, since NT_s and NT_r are also sent to the BoB before the deadline defined in Algorithm 5.1 (see ln 08 of Algorithm 5.1), at least one of the $f + 1$ nodes in the BoB will submit NT_s and NT_r to corresponding BCs. According to the liveness property, even if censorship attack is launched, NT_s and NT_r are guaranteed to be stored in corresponding BCs. As a result, successful state transitions will be achieved. However, if at least one participator (the sender and/or receiver) is malicious, it may launch a withholding attack (see lns 24 and/or 25 of Algorithm 5.1), or the balance in its account may not enough (see lns 28 and/or 29 of Algorithm 5.1). Then, Situation 3 will be triggered:

(iii) Situation 3 – the CCT is valid, and the sender and/or the receiver are malicious. According to lns 21-29 of Algorithm 5.1, each missed native transaction or proof transaction (due to withholding attacks), or the presence of valid fraud-proof transactions (due to the insufficient balance) will lead to a transfer of $Token_B$ for completing the unfinished token transfer operations. Thus, the state transition will be successful. As a result, the atomicity property is preserved.

Regarding the isolation property: the two native transactions will be executed concurrently in BC_i and BC_j, while the two proof transactions PT_s and PT_r will be

executed concurrently in the BoB. Due to the independence of BC_i and BC_j, each is unaware of the other's native transaction execution. Similarly, the execution of PT_s and PT_r in the BoB is independent – that is, the verification and execution of PT_s do not require any information from PT_r, and vice versa. Therefore, the isolation property is preserved.

To ensure the durability property, (i) the native transaction (such as NT_s) can only be executed after the CCT has been stored in the BoB and (ii) the proof transaction (such as PT_s) can only be executed after the native transaction NT_s are stored in BC_i. To ensure (i), if NT_s is stored in BC_i before the CCT has been stored in the BoB, i.e., $e_i^{'} \leq e_i + \bar{\kappa}$ (see ln 16 of Algorithm 5.1), PT_s will be discarded according to ln 18 of Algorithm 5.1. To ensure (ii), lns 06 and 18 also guarantee that if NT_s is not stored in BC_i (i.e., $Val_{proof}(P_s, NT_s, R\left(BH_i^{e_i^{'}}\right) \neq 1)$, PT_s will be discarded according to ln 18 of Algorithm 5.1.

When CCT_V is required to be sent to the BoB (see ln 22 of Algorithm 5.1), to ensure the durability property, CCT_V can only be executed after the deadline defined in ln 08 of Algorithm 5.1. The checking operation in ln 21 of Algorithm 5.1 guarantees that if CCT_V is sent to the BoB before the deadline, it will be discarded.

When a fraud transaction containing a Merkle proof is sent to the BoB (see lns 28 and 29 of Algorithm 5.1), the BoB will verify that whether the Merkle proof can attest that the corresponding native transaction (NT_s or NT_r) failed to be stored on its target BC because the on-chain balance of its sender's account fell below the required threshold. As a result, the fraud-proof transaction can only be executed when the Merkle proof is valid. Therefore, every transaction in Π_{CCTE} is executed following the criteria defined in the durability property. □

Theorem 5.3. *every native transaction in the CCTE process defined by* Π_{CCTE} *preserves the fungibility property.*

Proof. In each external BC, only native transactions that carry token transfer instructions will be executed. Since the instructions include only the sender's and the

receiver's IDs, the token type and the token transfer amount, they cannot be distinguished from standard token transfer transactions. Therefore, the fungibility property is preserved. □

5.5 Cross-Chain Data Interoperability Protocol

The proposed cross-chain data interoperability CCDI protocol Π_{CCDI} operates under the BoB-MBS including the BoB and N external blockchains BCs and defines a more generic CCDI process $\mathcal{P}_{CCDI}$. Without loss of generality, we consider that X user nodes register to participate in $\mathcal{P}_{CCDI}$. Each user node has a unique state data item, and the x-th $(x \in [X])$ user node (denoted as U_x) is from BC_i, where $i \equiv x \bmod N$. Additionally, Π_{CCDI} defines that each BC configures a relay group that comprises of $(f+1)$ relay nodes randomly selected from the M consensus nodes in the BoB. Π_{CCDI} further defines 3 smart contracts for $\mathcal{P}_{CCDI}$. SC_C and $SC_{\mathcal{F}}$ are deployed in the BoB, and SC_{BG} is deployed in every external blockchain BC_i $(i \in [N])$. Their functionalities are shown in Table 5-2.

TABLE 5-2. Smart contracts used in Π_{CCDI}

Notation	Functionality
SC_{BG}	Deploy battle games, create and update state data items
SC_C	Validate cross-chain transactions
$SC_{\mathcal{F}}$	Implement CCDI applications

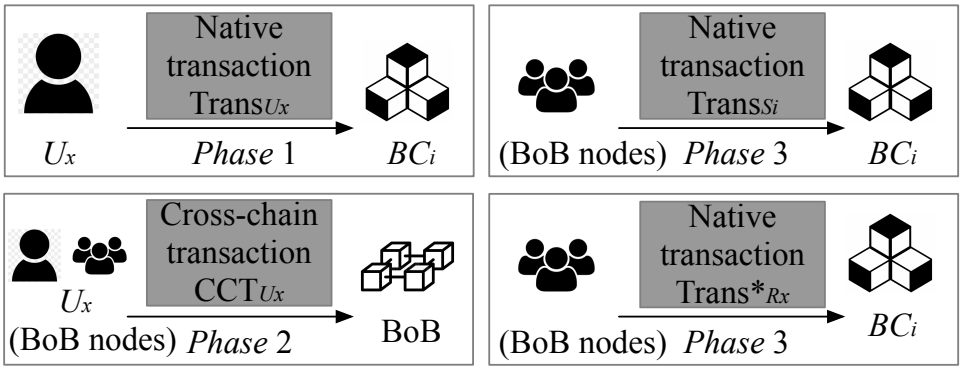

FIGURE 5-2. The transactions of Π_{CCDI} in each phase.

To illustrate a concrete CCDI application, we take battle games as an example. A battle game includes a Player versus Player (PvP) instance, which represents the scenario that two players battle with each other. A player uses a unique state data item representing its game character, which includes a role name, an attack value, and an experience value. In a PvP, a player's score is the sum of its character's attack and experience values. The player with a higher score will win the battle. These logics of battle games will be coded in the smart contracts SC_{BG} and $SC_{\mathcal{F}}$. In the following, we further present two building blocks of the CCDI protocol Π_{CCDI}.

5.5.1 Two Building Blocks of Π_{CCDI}

The first building block is an abstract CCDI application smart contract $SC_{\mathcal{F}}$ that can host any types of CCDI applications. In this paper, $SC_{\mathcal{F}}$ deploys the logic of battle games. To invoke $SC_{\mathcal{F}}$, a set of state data items' metadata (denoted as $\{SD_1, \dots, SD_x, \dots\}$) inputs into $SC_{\mathcal{F}}$ which outputs a new set $\{SD_1^{new}, \dots, SD_x^{new}, \dots\}$ according to the logic of battle games:

$$\{SD_1^{new}, \dots, SD_x^{new}, \dots\} = SC_{\mathcal{F}}(SD_1, \dots, SD_x, \dots) \quad (5.4)$$

where $\{SD_1^{new}, \dots, SD_x^{new}, \dots\}$ is the updated versions of $\{SD_1, \dots, SD_x, \dots\}$. Since $SC_{\mathcal{F}}$ only consumes metadata, the data format unification mechanisms [204] can be applied to the received metadata, harmonising their data formats before submitting them to $SC_{\mathcal{F}}$. Upon outputting the updated metadata, they can be converted back to their original data formats.

The second building block is a threshold-signature-based "transfer and in place data update" mechanism. Specifically, the nodes of the BoB collectively generate a threshold public key PK, and the i-th group generates an account using PK in the blockchain BC_i ($i \in [N]$). The smart contract SC_{BG} then grants PK the ability to update the state data items that PK possesses. Thus, PK now can invoke SC_{BG} to update its state data items. The details of the "transfer and in place data update" mechanism are as below.

When a user node U_x in the blockchain BC_i registers to participate in a CCDI process $\mathcal{P}_{CCDI}$, it first sends its state data item (denoted as SD_x) to the threshold public key PK, which acts as the locking operation. When SD_x is updated in the BoB through the smart contract $SC_{\mathcal{F}}$, the nodes in the BoB generate a transaction containing a threshold signature TS and the updated SD_x (denoted as SD_x^{new}). The transaction is then sent to BC_i to invoke smart contract SC_{BG}. If PK in SC_{BG} validates TS, SD_x is updated to SD_x^{new} in BC_i and sent back to U_x's account.

Next, we present the details of the protocol Π_{CCDI} (as shown in Figure 5-2) as below.

5.5.2 Phase 1 – Preparation

Let the CCDI process be initiated at the beginning of the BoB's epoch indexed by e_B and a deadline is set to be the end of the epoch $e_B + \kappa \times Max(\kappa_i)$, $(i \in [N])$, where κ is a secure parameter to ensure that every external blockchain BC_i has passed κ_i epochs since the beginning of epoch e_B. In epoch e_B, every user node U_x $(x \in [X])$ then sends a native transaction $Trans_{Ux}$ to the threshold public key PK in BC_i:

$$Trans_{Ux} =< SD_x, PK_x^i, PK, SC_{BG} > \tag{5.5}$$

where PK_x^i represents the account address of U_x in BC_i. BC_i checks whether PK_x^i owns the state data item SD_x. If not, $Trans_{Ux}$ is discarded; otherwise, BC_i transfers SD_x to PK and creates a key-value pair $< H(Trans_{Ux}), (SD_x, PK_x^i) >$ in the smart contract SC_{BG}, where $H(\cdot)$ is a hash function.

5.5.3 Phase 2 – Execution

If $Trans_{Ux}$ is stored in BC_i before epoch $e_B + \kappa \times Max(\kappa_i)$, a Merkle proof (denoted by P_x) and a cross-chain transaction (denoted as CCT_{Ux}) are generated:

$$CCT_{Ux} =< Trans_{Ux}, BC_i, P_x, SC_C > \tag{5.6}$$

The user node U_x and the $f + 1$ relay nodes in the i-th relay group $(i \in [N])$ send CCT_{Ux} to the BoB to invoke the smart contract SC_C. After receiving CCT_{Ux},

SC_C checks whether it has stored the state data item SD_x that is contained in the $Trans_{Ux}$'s field of CCT_{Ux}. If yes, CCT_{Ux} is discarded; otherwise, SC_C checks:

- Whether the index of the BoB's current epoch (denoted as e_B^c) satisfies: $e_B^c < e_B + \kappa \times Max(\kappa_i)$.
- Whether P_x can prove that $Trans_{Ux}$ has been immutably stored in BC_i.

If CCT_{Ux} passes all the 2 checking operations, the state data item SD_x contained in the cross-chain transaction CCT_{Ux} is stored in the smart contract $SC_{\mathcal{F}}$; otherwise, CCT_{Ux} is discarded. As shown in Figure 5-3, $SC_{\mathcal{F}}$ performs one of the two operations:

Case 1: If all the X cross-chain transactions are stored in the smart contract $SC_{\mathcal{F}}$ by epoch $e_B + \kappa \times Max(\kappa_i)$, when $SC_{\mathcal{F}}$ stores the X-th state data item, it updates the X state data items according to Eq. (5.4) and the updated SD_x is denoted as SD_x^{new}. For the set of C updated state data items $\boldsymbol{S}_i = \{SD_{x_1}^{new}, \dots, SD_{x_c}^{new}, \dots, SD_{x_C}^{new}\}$ from the blockchain BC_i ($c \in [C]$ and $\forall c, i \equiv x_c \mathrm{mod} N$), the M nodes of the BoB produce the threshold signature TS for the set $\boldsymbol{S}_i$.

Case 2: If there are some state data items among the X state data items not stored in the smart contract $SC_{\mathcal{F}}$ by epoch $e_B + \kappa \times Max(\kappa_i)$, $SC_{\mathcal{F}}$ identifies the number of the user nodes' state data items that have been received before epoch $e_B + \kappa \times Max(\kappa_i)$ (denoted as Y). If $Y > 1$, $SC_{\mathcal{F}}$ updates the Y state data items through (4). For the set of D updated state data items $\overline{\boldsymbol{S}}_i = \{SD_{x_1}^{new}, \dots, SD_{x_d}^{new}, \dots, SD_{x_D}^{new}\}$ from the blockchain BC_i ($d \in [D]$ and $\forall d, i \equiv x_d \mathrm{mod} N$), the M nodes of the BoB produce the threshold signature TS for the set $\overline{\boldsymbol{S}}_i$. If $Y \leq 1$, the M nodes of the BoB produce a threshold signature for the only received state data item.

If a valid cross-chain transaction CCT_{Ux} ($x \in [X]$) is sent to the BoB after epoch $e_B + \kappa \times Max(\kappa_i)$, the M nodes of the BoB also produce TS for the state data item SD_x contained in CCT_{Ux}, ensuring its unlocking during Phase 3 (see Case 2*).

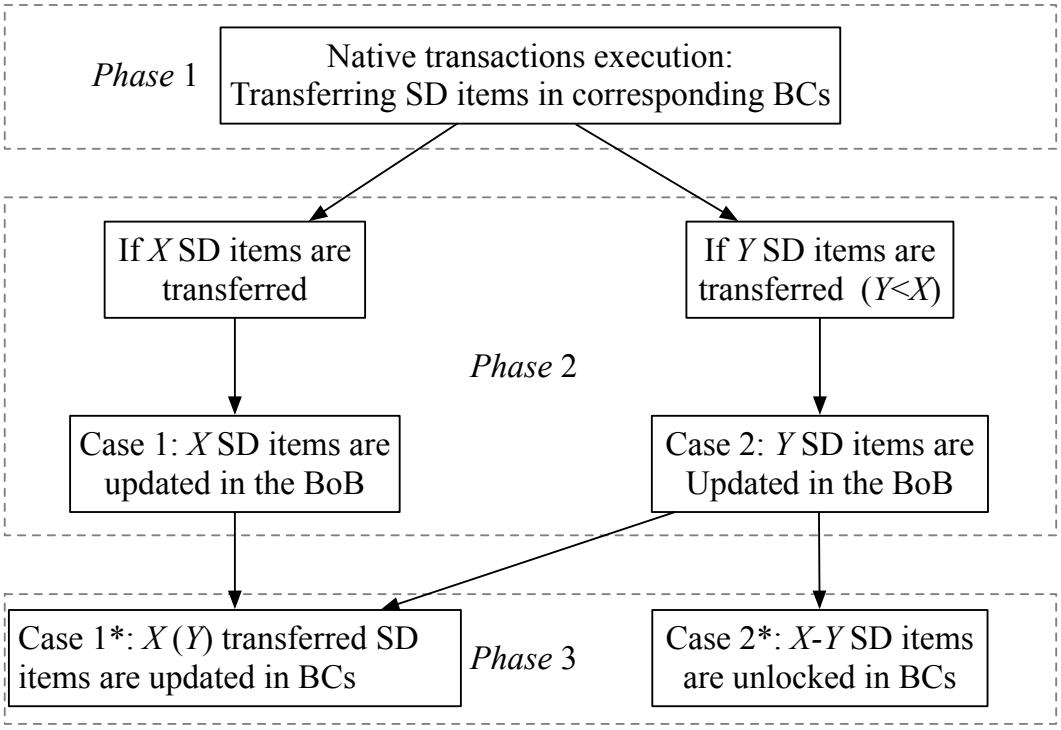

FIGURE 5-3. Flowchart of $\mathcal{P}_{CCDI}$.

5.5.4 Phase 3 – Settlement

The transferred state data items in Phase 1 are updated in their blockchains in this phase:

Case 1*: If the set $\boldsymbol{S}_i$ is generated in the BoB, a native transaction $Trans_{\boldsymbol{S}_i}$ is generated:

$$Trans_{\boldsymbol{S}_i} =< \boldsymbol{S}_i, TS, \boldsymbol{S}_{x_c}, SC_{BG} > \tag{5.7}$$

where $\boldsymbol{S}_{x_c}$ is a set containing the C hash values $\{H(Trans_{Ux_1}), \dots, H(Trans_{Ux_C})\}$.

After that, the $f+1$ relay nodes in the i-th group send $Trans_{\boldsymbol{S}_i}$ to the blockchain BC_i to invoke the smart contract SC_{BG}. SC_{BG} checks:

- Whether it has the key value pairs $< H(Trans_{Ux_1}), \left(SD_{x_1}, PK_{x_1}^i\right) >, \dots, < H(Trans_{Ux_C}), (SD_{x_C}, PK_{x_C}^i) >$.
- Whether the threshold public key PK validates TS.

If $Trans_{S_i}$ passes all the two checking operations, $Trans_{S_i}$ is regarded as valid; otherwise, $Trans_{S_i}$ is discarded.

If $Trans_{S_i}$ is valid, PK invokes SC_{BG} to update SD_{x_c} to $SD_{x_c}^{new}$ and sends $SD_{x_c}^{new}$ to the account address $PK_{x_c}^{i}$ ($c \in [C$]). After that, SC_{BG} deletes $< H(Trans_{Ux_c}), (SD_{x_c}, PK_{x_c}^{i}) >$.

The above procedures are also applied to $\overline{S}_i$ in Case 2.

Case 2*: To respond to a valid cross-chain transaction CCT_{Ux} in Case 2, a native transaction $Trans_{Rx}^{*}$ is generated:

$$Trans_{Rx}^{*} =< SD_x, TS, H(Trans_{Ux}), SC_{BG} > \quad (5.8)$$

The procedures for $Trans_{S_i}$ is then applied to $Trans_{Rx}^{*}$ and SD_x will be directly sent back to U_x's account PK_x^i.

5.5.5 Attacks Resolution Mechanism

According to Case 2, censorship and withholding attacks can cause that some state data items are updated while others are not updated, which compromises the atomicity property. To counter the two attacks, we propose an attacks-resolution mechanism which consists of S_{SDM}, a state-data migration scheme, and S_{MUE}, a malicious user nodes elimination scheme. S_{SDM} enables a user node U_x to migrate its state data item SD_x in a censorship-prone blockchain BC_i into other censorship-resilient BCs (such as BC_j). Different from the state-of the-art "lock (or burn)-to-claim" schemes that sacrifice the independence property of involved BCs during data migration processes, Π_{SDM} achieves data migration while preserving the independence property of involved BCs through our "transfer and in place data update" design. On the other hand, S_{MUE} eliminates the user nodes who deliberately withhold their native transactions in Phase 1 to make Case 2 happen. Moreover, S_{MUE} determines the set of participators in the CCDI process at the end of Phase 2, allowing Π_{CCDI} to update honest participators'

state data items without sacrificing its atomicity property. The details of the two protocols are presented as follows.

State-data migration scheme S_{SDM}. The user node U_x executes Eq. (5.5) to lock its state data item SD_x. At the same time, U_x creates an account in the blockchain BC_j (denoted as PK_x^j) and generates a newly initialised state data item $\overline{SD_x}$, which is SD_x's initial version. U_x then executes Eq. (5.5) to lock $\overline{SD_x}$. Once SD_x and $\overline{SD_x}$ are locked in BC_i and BC_j, respectively, U_x generates two cross-chain transactions according to Eq. (5.6) and sends them to the BoB. If the two transactions are valid, the M nodes of the BoB collectively produce the threshold signature TS_i for the locked SD_x and TS_j for the locked $\overline{SD_x}$. Finally, the $f+1$ relay nodes in the i-th group sends the transaction defined in Eq. (5.7) to transit SD_x to $\overline{SD_x}$ in BC_i. In the meantime, the $f+1$ relay nodes in the j-th group sends the transaction defined in Eq. (5.7) to transit $\overline{SD_x}$ to SD_x in BC_j.

Supported by the scheme S_{SDM}, all user nodes can migrate their state data items to other censorship-resilient BCs, thereby mitigating the negative impacts of censorship attacks.

Malicious user nodes elimination scheme S_{MUE}. S_{MUE} further eliminates the user nodes who make Case 2 happen through withholding attacks. First, a deposit mechanism is adopted, which enforces that every user node needs to deposit a certain number of tokens in the BoB before registering to participating in a CCDI process $\mathcal{P}_{CCDI}$. If a user node does not fulfill the deposit operations before the state of $\mathcal{P}_{CCDI}$, the authority of the user in $\mathcal{P}_{CCDI}$ is canceled. Second, if Case 2 happens, the BoB identifies the user nodes who fail to store their state data items in the smart contract $SC_{\mathcal{F}}$ before epoch $e_B + \kappa \times Max(\kappa_i)$. The accounts of the identified user nodes are then removed from the X registered user nodes, and the deposits of the identified user nodes are forfeited. Lastly, denoting the identified number of user nodes be $X-Y$, the remaining Y user nodes are authorised for the CCDI process $\mathcal{P}_{CCDI}$.

Under the above design of the scheme S_{MUE}, the final authorised participators in

$\mathcal{P}_{CCDI}$ can only be determined at the end of Phase 2. In this way, more user nodes are allowed to participate in $\mathcal{P}_{CCDI}$ at the beginning of $\mathcal{P}_{CCDI}$, that aligns with complex cross-chain interaction applications in real-world scenarios. However, malicious user nodes can be identified and eliminated, guaranteeing the completion of the CCDI service for the remaining user nodes.

5.5.6 Security Analysis

This section is devoted to analysing the security of the proposed CCDI protocol. The analysis is represented by the following theorems:

Theorem 5.4. *For the CCDI process $\mathcal{P}_{CCDI}$ defined in Π_{CCDI}, the independence property of each participating BC_i ($i \in [N]$) in the BoB-MBS is preserved.*

Proof. In $\mathcal{P}_{CCDI}$, two types of native transactions are executed in BC_i (i.e., $Trans_{Ux}$ and $Trans_{\boldsymbol{S}_i}/Trans_{Rx}^{*}$):

(i) Executing $Trans_{Ux}$ involves checking the ownership of SD_x. As SD_x is the state data item of BC_i, the execution of this operation does not need the external state data.

(ii) The execution of $Trans_{\boldsymbol{S}_i}/Trans_{Rx}^{*}$ involves the two checking operations and the "in place data update" operation. The two checking operations require to use the state data PK and $\boldsymbol{S}_{x_c}$. To execute the "in place data update" operation, the state data, i.e., the locked SD_x, is required. Apparently, PK, $\boldsymbol{S}_{x_c}$, and the locked SD_x are the state data of SC_{BG}. Thus, the execution of $Trans_{\boldsymbol{S}_i}/Trans_{Rx}^{*}$ does not require any external state data except for that of BC_i, and proof completes. □

Theorem 5.5. *Under the corruption and network models defined in Section III-A, if every external BC_i ($i \in [N]$) and the BoB hold the safeness and liveness properties, by combining the attacks resolution mechanism, Π_{CCDI} will hold the atomicity, isolation, durability properties.*

Proof. To demonstrate the atomicity property, we analyse the state transitions of

$\{SD_1, \ldots, SD_X\}$ through different phases:

At Phase 2, with the support of the $f+1$ relay nodes, as long as a state data item (e.g., SD_x) is immutably stored in corresponding blockchain (e.g., BC_i) before epoch $e_B + \kappa \times Max(\kappa_i)$, at least one of the $f+1$ relay nodes will send the corresponding cross-chain transaction (e.g., CCT_{Ux}) to the BoB. Therefore, there are three possible situations. $\boldsymbol{S}_1$ (i.e., Case 1): X state data items are stored in $SC_{\mathcal{F}}$ before epoch $e_B + \kappa \times Max(\kappa_i)$. $\boldsymbol{S}_2$ (i.e., Case 2): only Y $(1 < Y < X)$ state data items meet the deadline. $\boldsymbol{S}_3$: only Y $(1 \geq Y)$ state data items meet the deadline.

At Phase 3, with the support of the $f+1$ selected relay nodes, at least one honest relay node will send valid native transactions to the blockchains for the updates of locked state data in Phase 1. Thus, there are three possible situations:

$\boldsymbol{S}_4$: all the honest nodes' state data items are eventually updated in corresponding blockchains.

$\boldsymbol{S}_5$: some of honest users' state data items are updated to their new state versions, while the other honest users' state data items are just unlocked and unchanged.

$\boldsymbol{S}_6$: all state data items are unlocked and unchanged.

With the scheme S_{SDM}, every user node can move to the censorship-resilient blockchains. Therefore, honest user nodes can store their state data items in the smart contract $SC_{\mathcal{F}}$ before epoch $e_B + \kappa \times Max(\kappa_i)$ for sure, which can potentially eliminate the situation of $\boldsymbol{S}_6$. Moreover, the malicious user nodes who withhold their native transactions in Phase 1 can be identified and eliminated by the scheme S_{MUE}, and the pre-defined X user nodes for the CCDI process can be updated to the Y user nodes. As a result, the remaining honest users must be a part of the Y user nodes, making $\boldsymbol{S}_5$ unable to happen. Thus, only $\boldsymbol{S}_4$ can happen and Π_{CCDI} holds the atomicity property.

Regarding the isolation property, in Phase 1 and Phase 3, each involves the concurrent execution of X native transactions. Phase 2 involves the concurrent execution of X cross-chain transactions.

In Phase 1, the native transactions are associated with the operations of ownership

checking for state data items and locking state data items. For the native transaction $Trans_{Ux}$, checking and locking U_x's state data SD_x only needs to check U_x's account information, that does not interfere with the execution of $Trans_{Uy}$ $(x \neq y)$.

In Phase 2, the execution of the cross-chain transactions involves the two operations of checking time requirement and checking the validity of Merkle proofs. Firstly, checking the epoch index requirement is independent operation, as the epoch index used for checking will not be changed even if an epoch checking operation is conducted. Secondly, checking CCT_{Ux}'s Merkle proof does not rely on the information from CCT_{Uy}'s Merkle proof, and vice versa. As a result, the execution of CCT_{Ux} does not interfere with the execution of CCT_{Uy}.

In Phase 3, the execution of $Trans_{\boldsymbol{S}_i}/Trans^*_{Rx}$ is associated with checking the validity of the received threshold signature, checking key value pairs, and performing "in place data update" operations. Firstly, checking the validity of the threshold signature TS or the key value pairs is an independent operation and cannot change any state after the checking operation. Secondly, updating SD_x does not interfere with burning SD_y, as they are different state data items. As a result, the execution of $Trans_{\boldsymbol{S}_i}$ or $Trans^*_{Rx}$ does not interfere with the execution of $Trans_{\boldsymbol{S}_j}$ with $i \neq j$.

As for the durability property, Π_{CCDI} involves three orderly executed transactions: the cross-chain transaction CCT_{Ux} can only be executed after the native transaction $Trans_{Ux}$ is stored in BC_i; and the native transaction $Trans_{\boldsymbol{S}_i}/Trans^*_{Rx}$ can only be executed after CCT_{Ux} is stored in the BoB. With the support of the Merkle proof scheme, if $Trans_{Ux}$ is not stored in BC_i, its valid Merkle proof cannot be generated. As a result, the BoB will not execute CCT_{Ux}. Supported by the threshold signature scheme, if CCT_{Ux} is not stored in the BoB, a valid threshold signature cannot be generated. As a result, BC_i will not execute $Trans_{\boldsymbol{S}_i}/Trans^*_{Rx}$. □

5.6 Simulation

5.6.1 Simulation Setup

For the simulation tool, Multiple Ethereum BCs (including the external BCs and the BoB) are created by Ganache [218]. We use Solidity scripts to code the smart contracts, which are deployed and tested in Remix-IDE Ethereum [219]. Merkle tree and Merkle proofs are written by Python. The complete process to conduct the simulation are well-presented in [227].

For the simulated parameters, the block generation time of the Ethereum BC, the size of a block header, a block and a normal token transfer transaction (without considering the field of a transaction's signature with 48 bytes) are set to be 15 seconds, 510 bytes, 8×10^4 bytes and 50 bytes, respectively. The above settings align with the setting of the real-world Ethereum BC [228]. Additionally, we use SHA-256 hash function (32 bytes for each hash value) and the Merkle proof depth is fixed at 5. The threshold signature scheme is instantiated over the BLS12-381 pairing-friendly elliptic curve. The security parameter κ_i for BC_i ($\forall i \in [N]$) is set to be k.

The system is tested subjected to four metrics:

(i) *Communication overhead* $Comm_o$ – the size of transactions (in bytes) required for executing a CCDI process.

(ii) *Computation overhead* $Comp_o$ – the computation (in gas) required for executing a CCDI process.

(iii) *Memory overhead* M_o – the memory (in bytes) required for executing a CCDI process.

(iv) *Latency* L_o – the time slot (in seconds) between the beginning of a CCDI process and the state data items of honest user nodes involved in the CCDI process all being updated.

5.6.2 Evaluation of Π_{CCTE} without Attacks

We test the execution of a CCTE process with the proposed Π_{CCTE} without considering attacks. The total communication is 330 bytes, in which a cross-chain transaction with 230 bytes and two native transactions, each with 50 bytes, are included. The gas cost of the smart contracts is reported in Table 5-3. The memory cost is $k \times 510 \times N$ bytes, all undertaken by the BoB. When the external BCs use deterministic protocols, the latency of the CCTE process is 30 seconds (15 seconds are spent on the BoB and 15 seconds are on the BCs); when the protocols are probabilistic, the latency is $30+15 \times k$ seconds.

We compare the proposed Π_{CCTE} and two existing CCTE schemes without considering attacks: a cryptocurrency exchange protocol (denoted as Π_{DCE}) [185] and an adaptor-signature-based protocol (denoted as Π_{AS}) [192]. Figure 5-4 shows the comparison of the communication cost under the 3 protocols. It shows that although the Π_{AS} protocol is with the least communication cost (consuming 100 bytes), the proposed protocol is also efficient, requiring only 330 bytes to process a CCTE process. The communication cost of Π_{DCE} is higher than that of Π_{CCTE} due to its requirement for 1 cross-chain transaction, 3 token transfer transactions and $2C$ validation transactions, where C is the number of committee nodes.

TABLE 5-3. Smart contracts gas cost of Π_{CCTE}.

Notation	Deployment gas cost	Execution gas cost
SC_D	2,901,508	-
SC_T	1,561,365	428,991
SC_{AMM}	7,610,008	-
SC_{MPV}	1,270,187	39,167

"-"means the gas costs of the smart contracts are covered by SC_T. The gas costs of SC_T and SC_{MPV} are caused by executing a valid cross-chain transaction and a valid proof transaction, respectively.

As shown in Table 5-3, Π_{CCTE} consumes 512,911 (470,911 for a CCT and 42,000 for two token transfers) gas to execute a CCTE process, which is less than Π_{DCE} (440,525+163,780*C) gas [185], while higher than Π_{AS} (42,000 gas). Π_{CCTE} incurs higher gas costs than Π_{AS} because Π_{AS} executes CCTs off-chain, while Π_{CCTE} executes CCTs on-chain. Since CCTs run within the BoB, this overhead can be eliminated by implementing the BoB with a gas-free constructions [229, 230].

Since Π_{AS} does not need a BoB, Figure 5-5 only shows the comparison of the memory cost between Π_{CCTE} and Π_{DCE}. Π_{CCTE} requires the BoB nodes to store a minimum number (i.e., k) of block headers per external BC to verify Merkle proofs. In contrast, Π_{DCE} requires the committee nodes of its BoB to store the latest k blocks of the N BCs, which results in higher memory cost especially when N continually increases. As a result, our protocol provides a scalable solution for the implementation of the BoB, as every consensus node in the BoB only caches kilobytes-level sized data for the verification tasks during CCTE processes.

Figure 5-6 shows the latency cost comparison under the 3 protocols. Π_{CCTE} uses only 2 phases to complete the CCTE process, which is the same as that of Π_{AS}. Moreover, when the BCs adopt probabilistic consensus protocols and $k = 2$, our protocol Π_{CCTE} reduces its latency by 52% compared to Π_{AS}. This is due to the parallelised design of Π_{CCTE}, enabling the token transfer transactions to be executed concurrently in Phase 2. As a result, this parallelised design not only improves protocol latency but also eliminates the censorship attack surface that originates from sequential token transfers in [185] and [192].

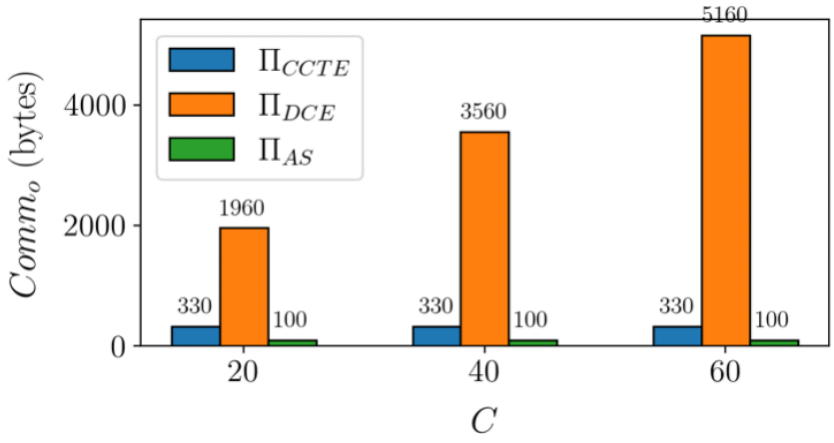

FIGURE 5-4. Communication cost comparison in CCTE. Both Π_{CCTE} and Π_{AS} does not introduce the variance.

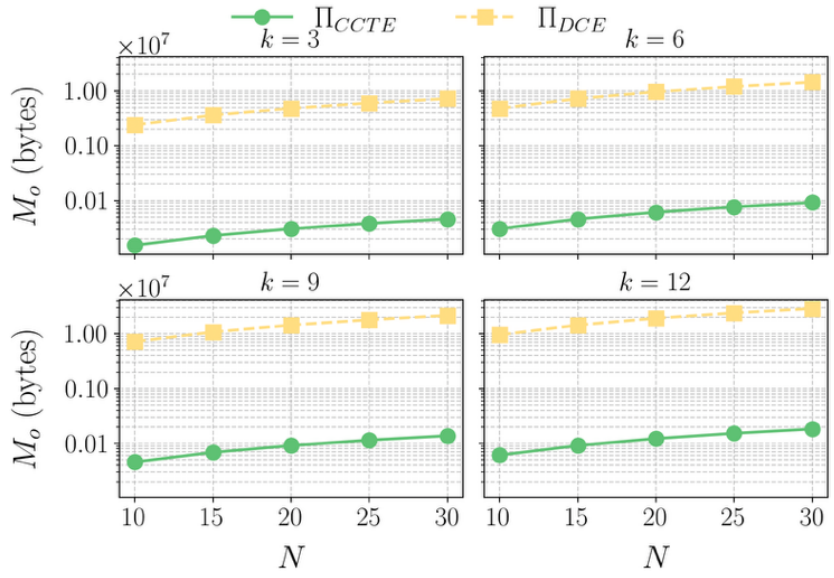

FIGURE 5-5. Memory cost comparison in CCTE under varying external BC numbers N and k.

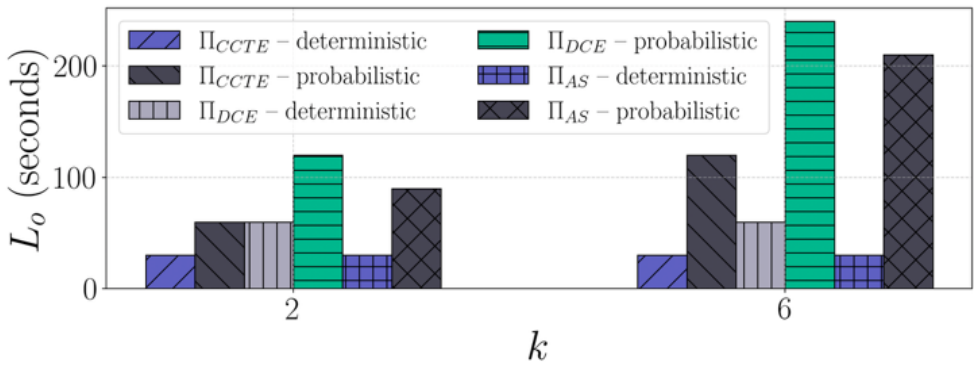

FIGURE 5-6. Latency cost in CCTE. “deterministic” and “probabilistic” denote the consensus types.

TABLE 5-4. Comparison on the four metrics of the three protocols in a CCTE process under attacks.

Types	$Comm_o$	$Comp_o$	M_o	L_o
Π_{CCTE}	980	732,251	510kN	45+15k
Π_{DCE}	410+40C	419,525+163,780C	8kN×10^4	45+15k
Π_{AS}	50	21,000	0	30+30k

A Proof-of-Work transaction in [185] is set to be 40 bytes in this paper.

5.6.3 Evaluation of Π_{CCTE} under Attacks

Table 5-4 summarises the results of the four key metrics for the 3 protocols under the censorship and withholding attacks. In the worst-case (both sender and receiver malicious), Π_{CCTE} incurs 980 bytes per CCTE operation (including two extra 210-byte proof transactions plus one extra 230-byte cross-chain transaction CCT_V). Consequently, its total gas consumption rises to 732,251 (219,340 gas attributable to those three transactions) and its latency increases by one additional block interval (from 30+15k seconds to 45+15k seconds). Its memory overhead, however, remains unchanged.

Although extra cost is introduced, Π_{CCTE} ensures its atomicity property and completes the two token transfers. In contrast, Π_{AS} fails to hold the atomicity property under the attacks, while Π_{DCE} rolls back all operations in the CCTE process. Thus, our protocol is more robust and practical under the adversarial scenarios.

5.6.4 Evaluation of Π_{CCDI} without Attacks

By implementing battle games (see Section 5.5) in $SC_{\mathcal{F}}$ to act as the CCDI application, we compare the proposed Π_{CCDI} with other 2 protocols named as MAP [210] and PoS-Co [208]. For simplicity, we consider that each BC has $\lfloor X/N \rfloor$ state data items participating in a CCDI process. The following results are derived without considering attacks. Figure 5-7 compares the communication cost of the 3 protocols. It can be seen that our proposed protocol Π_{CCDI} reduces communication cost by at least 35% compared to the two protocols. This is because that in Π_{CCDI}, only $2X$ transactions

(see Eq. (5.5) and Eq. (5.6)) and N native transactions (see Eq. (5.7)) are required for updating X state data items. In contrast, PoS-Co requires $3X+1$ transactions to update X state data items. MAP's performance in communication cost is worst.

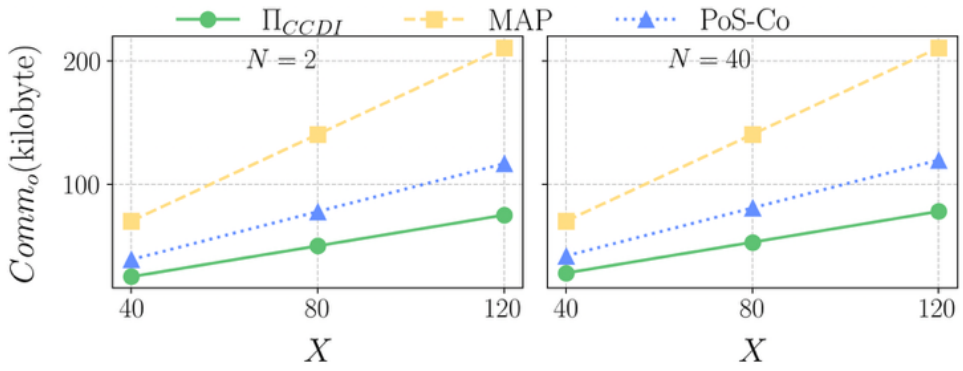

FIGURE 5-7. Communication comparison in CCDI under varying X and N.

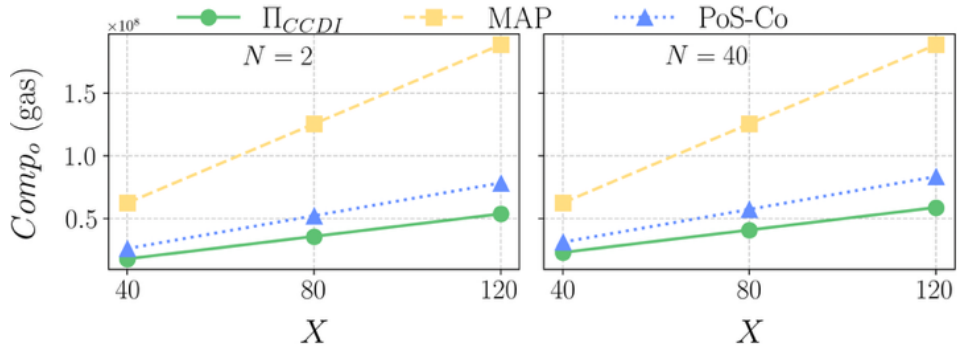

FIGURE 5-8. Gas comparison in CCDI under varying X and N (without considering the data storage costs of MAP and PoS-Co for validating proofs).

TABLE 5-5. Gas consumption of basic operations

Operation types (required protocols)	Gas cost	Price
Lock a state data item (MAP and PoS-Co)	186,443	$3.7
Transfer a state data item (Π_{CCDI})	59,963	$1.2
Verify a MP (Π_{CCDI} and PoS-Co)	39,167	$0.8
ZK-SNARK (MAP)	250,000 [28]	$5
Verify a time metric (all protocols)	20,533	$0.4
Store a state data item (Π_{CCDI})	118,645	$2.4
PvP battle games (all protocols)	63,779	$1.3
Verify a TS (Π_{CCDI} and PoS-Co)	113,000	$2.3
Update a state data item (all protocols)	51,244	$1
Mint a state data item (MAP and PoS-Co)	265,479	$5.3

The price of ETH is considered as $2,000 in this calculation.

Table 5-5 shows the gas consumption of the operations involved in the 3 protocols. Building upon Table 5-5, Figure 5-8 shows the comparison results of gas cost under the 3 protocols. It can be observed that Π_{CCDI} reduces gas consumption by at least 41% compared to the two protocols when $X = 120$, $N = 40$. Π_{CCDI}'s gas efficiency stems from three key design choices. First, in Phase 2, Π_{CCDI} records only each state data item's metadata in $SC_{\mathcal{F}}$, rather than minting and storing a state data item on the BoB. Table 5-5 shows that metadata storage requires far less gas than creating new state data items. Second, in phase 3, a single threshold signature lets Π_{CCDI} update $\lfloor X/N \rfloor$ state data items in one BC at once, incurring a fixed gas cost of 21,000+113,000+51,244X/N units. In contrast, MAP and PoS-Co needs one transaction to update one state data item. Third, Π_{CCDI} only needs 3 phases to complete a CCDI process, while MAP and PoS-Co need 5 and 4 phases to complete a CCDI process, respectively. All the three merits make Π_{CCDI} \$2,593 and \$493 cheaper than MAP and PoS-Co, respectively, when $X = 120$ and $N = 40$.

Figure 5-9 shows the comparison results of memory cost. The memory cost of Π_{CCDI} is $N \times k \times 510$ bytes, which is the same as Π_{CCTE}, as the two protocols ensure that every BC holds the independence property. Notably, the $N \times k \times 510$ bytes memory costs are only consumed by the BoB of the other two protocols. However, as MAP and PoS-Co do not hold the independence property, the BCs in their constructions consumes additional memory costs. In MAP, each BC needs to store the last k block headers of the BoB and a set of validators' public keys. PoS-Co is more efficient than MAP since each BC only requires updating a threshold public key within a certain number of epochs. However, for every CCDI application, each BC needs to store k latest root hashes of the BoB. As a result, Π_{CCDI} reduces its memory burden by at least 12% compared to the two protocols when $k = 16$, $N = 40$. According to [228], storing 1 kilobytes data consumes 640,000 gas, which makes Π_{CCTE} \$1,922 and \$560 cheaper than MAP and PoS-Co, when $N = 40$, $k = 16$.

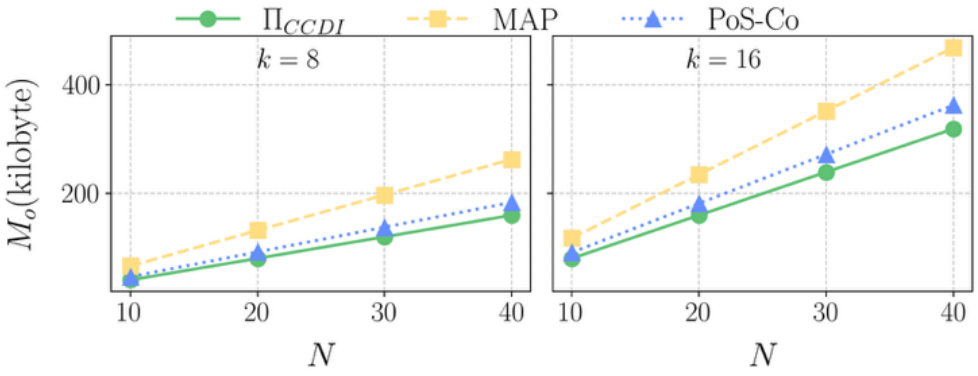

FIGURE 5-9. Memory comparison in CCDI under varying N and k (considering 40 validators in MAP's PoS consensus).

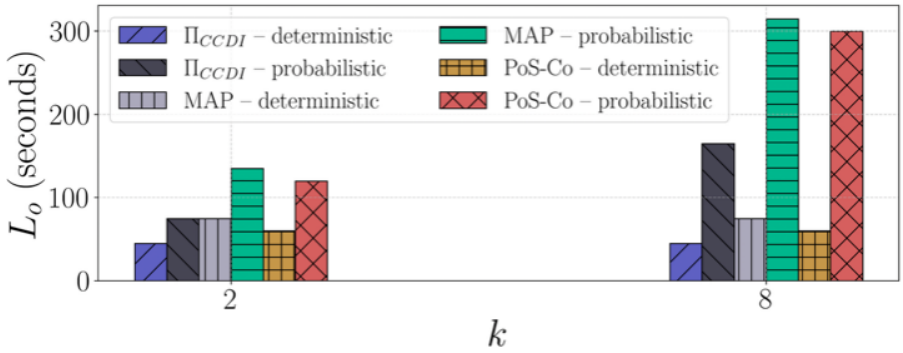

FIGURE 5-10. Latency comparison in CCDI. "deterministic" and "probabilistic" denote the consensus types.

Figure 5-10 shows the comparison result of latency cost. Since Π_{CCDI} only consumes 3 phases to complete a CCDI process, while MAP and PoS-Co require 5 and 4 phases, respectively, Π_{CCDI} reduces its latency by at least 33% compared to other two protocols. The efficiency of Π_{CCDI} is due to the "transfer and in place data update" design, which enables direct updates on multiple locked state data items using a single transaction within a single phase. By reducing latency, Π_{CCDI} narrows adversaries' window for mounting attacks, thereby enhancing system's robustness.

5.6.5 Evaluation of Π_{CCDI} under Attacks

We evaluate the 3 protocols under both censorship and withholding attacks. Table 5-6 summarises the four key metrics for each protocol. As shown, Π_{CCDI} outperforms both MAP and PoS-Co across every metric. This efficiency arises from two core mechanisms.

TABLE 5-6. Comparison on the four metrics of the three protocols in a CCDI process under attacks

Types	$Comm_o$	$Comp_o$	M_o	L_o
Π_{CCDI}	640Y+80N	416,032Y+63,779(Y-1)+134,000N	510kN	45+15k
MAP	1824Y	1,543,166Y	510kN+(1,444+150k)N	>75+30k
PoS-Co	996Y+80N	621,500Y+134,000N	574kN+96N	>60+30k

In the CCDI process, we consider $X - Y$ malicious user nodes who can launch withholding attacks.

First, the state-data migration scheme (S_{SDM}) allows honest user nodes to migrate their state data items to censorship-resilient BCs, completely neutralising censorship attacks; by contrast, MAP and PoS-Co must resort to timeout-based mechanisms to roll back an entire CCDI process for the preservation of their atomicity property. Second, the malicious user nodes elimination scheme (S_{MUE}) eliminates malicious user nodes and allows the remaining user nodes to continue the current CCDI process. Consequently, even if these malicious user nodes launch withholding attacks, the state data items of the remaining honest user nodes are still updated, unlike MAP and PoS-Co that roll back all operations involved in the CCDI process.

5.7 Conclusions and Future Work

This chapter proposes CCTE and CCDI to support cross-chain data interoperability in BoB-MBS. CCTE supports cross-chain token exchange while preserving the independence of participating BCs. By allowing parallel execution of cross-chain token-transfer transactions, CCTE reduces confirmation latency relative to sequential time-lock-based schemes under the evaluated settings. Its trade-off is higher communication, computation and memory cost than time-lock-based schemes [189-192]. The security analysis shows that CCTE preserves atomicity and completes token exchange under the withholding and censorship attacks considered in this chapter.

CCDI addresses cross-chain SD item updates through the threshold-signature-

based “transfer and in place data update” mechanism. The protocol updates selected SD items in participating BCs without requiring those BCs to give up their independence. Because “in place data update” requires one transaction-execution phase in the target BC, CCDI has lower latency than the compared state-of-the-art protocols under the evaluation settings in Section 5.6. The state-data migration scheme and malicious user-node elimination scheme are used to handle withholding and censorship attacks.

The BoB-MBS assumption in this chapter is not that external BCs blindly trust each other. Instead, each external BC keeps local verification. An external BC accepts a CCTE or CCDI transaction only after its smart contract verifies the required evidence, such as a Merkle proof or a TS under the threshold public key. This model is suitable for demand-side energy systems in which operational records may be maintained by different but interacting operators, including prosumer communities, aggregators, virtual power plants and market platforms. In multi-microgrid P2P energy trading, separate microgrids may maintain separate energy BCs while still needing cross-chain trading for joint operation [294]. Demand response, EV coordination and decentralised energy management also require transparent coordination among many demand-side actors [295]. If these actors maintain records on different BCs, CCDI can update selected SD items, such as user status, flexibility records or settlement metadata, without requiring every BC to store all other BCs’ block headers. Similarly, electricity, carbon and TGC markets may be operated separately but coupled by market rules; a cross-chain day-ahead market has been proposed for this coupled electricity-carbon-TGC setting [296]. Therefore, the protocols in this chapter provide theoretical support for demand-side applications that require cross-chain token exchange or selected SD item updates while preserving local smart-contract verification, atomicity, isolation and durability.

Future research can proceed in three directions. First, CCTE should reduce its reliance on the deposit mechanism while preserving atomic exchange and the ability to complete token exchange under withholding and censorship attacks. Second, CCDI should reduce communication cost by using more efficient message-sending or leader-

based mechanisms, while retaining local verification by participating BCs. Third, BoB-MBS should be evaluated in concrete demand-side energy applications with independently maintained BCs, such as multi-microgrid trading, aggregator-community flexibility exchange and coupled electricity-carbon-TGC markets.

CHAPTER 6

Distributed Optimisation-based Peer-to-Peer Energy Market Clearence

Distributed optimisation (DO) algorithms are cornerstones of modern peer-to-peer (P2P) energy market clearing. Current privacy preserving DO algorithms, however, are computationally expensive and vulnerable to Byzantine attacks that can prevent their optimal convergence.

This paper addresses these challenges by proposing a DO-algorithm-agnostic framework that leverages Trusted Execution Environments (TEEs) and consists of two components. For the first component, a D-TASK protocol is proposed to select TEEs that deploy the same predefined information. The D-TASK protocol establishes a foundation for a lightweight signature-based proof method used by the second component and constructs a private environment for executing any type of DO algorithms. For the second component, a TEAR-DO mechanism is designed to enable the execution of DO algorithms on plaintexts with minimal redundant computing and ensures all Byzantine attacks can be effectively detected and resolved, thereby ensuring optimal convergence of DO algorithms under Byzantine attacks.

Theoretical analysis demonstrates the effectiveness of the framework in preserving data privacy and defending against Byzantine attacks. Simulation results show that by comparing with a state-of-the-art privacy preserving scheme, the framework can achieve at least an order of magnitude speedup in completing a P2P energy market's clearing process. Because of its time-efficient, privacy-preserving and Byzantine-tolerant features, the framework may be of independent interest for a wide range of applications beyond P2P energy trading.

6.1 Introduction

The ever-growing proliferation of distributed renewable energy resources has been transforming energy entities from pure energy consumers to energy prosumers that are capable of generating and consuming energy simultaneously. This technical trend fosters the research and development of peer-to-peer (P2P) energy trading [231], which refers to as energy trading (in most cases, electricity) among prosumers.

A key task of P2P energy trading is to optimally match buyers' bids with sellers' offers subject to specific constraints. This process is called P2P energy market clearing. Since P2P energy trading systems are usually distributed systems comprising multiple prosumers, distributed optimisation (DO) techniques have been applied to clear the market. For instance: [232] applies a primal dual gradient method to solve a market clearing optimisation problem in a distributed manner. [233] develops a game-theoretic scheme that enables prosumers to strategically determine their energy trading plans to maximise the welfare. [234] develops an alternating direction method of multipliers-based DO scheme to speed up the process with reduced communications.

Among these research efforts, a DO algorithm usually decouples a centrally formulated optimisation problem into multiple subproblems that are processed by different executors; the executors iteratively exchange limited information with each other and mutually output to a global optimal solution [235]. By employing DO algorithms, single point of failure risk can be effectively mitigated [236]. There are three critical requirements influencing the applicability of DO algorithms in real-world P2P energy market clearing systems:

Privacy preserving requirement: The iterative problem-solving process in DO-based P2P energy market clearing approaches require prosumers exchanging coupling variables (e.g., bilateral traded energy) during each iteration [237]. Under this context, research has been demonstrated that an adversary, by observing these variables, can perform inference attacks to reconstruct sensitive information, such as prosumers' load profiles and confidential grid parameters [238].

Optimal convergence requirement: Achieving optimal convergence in an iterative DO process is crucial for maximising the social welfare of all participants. However, guaranteeing convergence becomes a nontrivial challenge in adversarial settings, where Byzantine nodes, fully controlled by an adversary, can exhibit arbitrary misbehaviour to prevent the iterative DO process from reaching its correct optimal solution [239].

Real-time operation requirement: P2P energy trading requires market clearing and settlement to be completed within certain intervals, e.g., the 5-minute windows [240]. However, this strict timeliness poses a significant challenge, as the computational and communicational overhead required to guarantee both data privacy and optimal convergence often conflicts with such rapid execution deadlines.

6.1.1 State-of-the-Art and Research Gaps

Research efforts have been devoted to providing the privacy preservation guarantees in P2P energy market clearing. One prominent approach is to combine DO algorithms with cryptographic techniques like homomorphic encryption (HE) and multi-party computation (MPC) [237], [241-243]. These technologies enable algorithmic operations to be performed directly over ciphertexts. However, the practicality of such cryptographic solutions for real-time energy market clearing (e.g., 5-minute intervals) is severely undermined by their prohibitive computational overhead, as operations on ciphertexts are orders of magnitude slower than operations on plaintexts [244]. Although studies like [237] achieve efficient computing on ciphertexts in terms of homomorphic addition and scalar multiplication, non-linear operations (e.g., homomorphic multiplication) remain computationally expensive. To circumvent this limitation, DO algorithms like primal-dual gradient methods are often redesigned to restrict all computations on sensitive data solely to linear operations [237, 241, 242]. However, these approaches restrict DO algorithms solely to first-order methods, whose slower convergence rates necessitate thousands of iterations, thereby causing excessive communication overhead.

Additionally, research has confirmed that Byzantine nodes can bias the convergence by injecting maliciously crafted ciphertexts [239]. To counter the threat of Byzantine attacks, some research has turned to blockchain (BC) technology [245]. The goal is to leverage BC's redundant computing and consensus mechanism to create a trustworthy environment for P2P energy trading. However, directly implementing DO algorithms on-chain is widely considered impractical [246, 247]. This is because every DO iteration would necessitate a network-wide consensus to validate updated variables, introducing a level of time delay that is fundamentally incompatible with the real-time requirements of P2P energy market clearing.

Recently, Trusted Execution Environments (TEEs) have emerged as a promising alternative for ensuring both data privacy and trustworthy computation [248, 249]. A TEE is a hardware-isolated area within a processor that protects the confidentiality and integrity of its resident data, rendering them inaccessible to others [250]. Crucially, a TEE can prove its computational trustworthiness through a process called remote attestation and this process often consumes 0.1 to 1 second [250]. Compared with HE, MPC and other technologies like zero knowledge proofs [251], TEEs are much more computational efficient as no algorithmic operations are executed on cyphertexts, and its remote attestation time is acceptable at most cases. However, the need to perform a remote attestation at each of the hundreds or thousands of iterations required in a DO process creates a prohibitive time latency, severely hindering TEEs' applicability for DO-based P2P energy market clearing.

In summary, the above analysis reveals a fundamental trilemma in P2P energy market clearing, as no existing approach simultaneously guarantees: (i) the robust privacy preservation for sensitive energy data; (ii) the optimal convergence of DO algorithms under Byzantine attacks; and (iii) the time efficiency required for real-time energy market clearing.

6.1.2 Contributions

This work is devoted to addressing the identified trilemma related to the P2P energy market clearing. This paper proposes a framework that utilises a BC and a cluster of TEEs to solve the trilemma. In the framework, participants first encrypt their sensitive data, such as bids and offers, and submit it to a BC where each BC node is equipped with a TEE. The BC then commits the encrypted data into blocks. Finally, by accessing the encrypted data in the blocks, the TEEs of the BC nodes collectively solve the market clearing optimisation problem.

This framework, however, still faces three fundamental challenges in resolving the trilemma. First, as the BC stores only encrypted data, a unified decryption scheme is required to enable every TEE to decrypt the ciphertexts and subsequently operate directly on the resulting plaintexts. Second, given that per-iteration remote attestation is prohibitively slow (at least hundreds of milliseconds in a distributed environment [250]), a key challenge is to establish computational trust among TEEs with lower latency. Third, under the scenario where TEEs may be controlled by Byzantine nodes, the time-efficient challenge is achieving optimal convergence under Byzantine attacks without introducing massive computational redundancy like BC-based solutions [245]. To solve the three challenges in the framework, a shared secret key *SKS* and a pre-defined binary script ($\mathbb{C}$) are introduced. *SKS* is the core to solve all the three challenges, while $\mathbb{C}$, including pre-defined codes and data, is the core to solve the last two challenges. This paper further designs a protocol and a mechanism for the proposed framework:

1. D-TASK protocol. To solve the first challenge, the shared secret key *SKS* and the pre-defined binary script $\mathbb{C}$ are securely deployed into each TEE. *SKS* corresponds to a public key *PKE*, allowing all participants to encrypt their data with *PKE* while enabling any TEE to decrypt it using *SKS*. For security purposes, deploying *SKS* and $\mathbb{C}$ into TEEs must satisfy the three properties when aiming to tackle the identified trilemma: 1) *SKS* must be only known to the selected TEEs for the privacy preservation

purpose. 2) No 3rd party can be relied on during the deployment process to avoid single point of failure. 3) the properties 1) and 2) can be achieved even in the presence of Byzantine nodes who can perform any possible attacks to their controlled TEEs. To fill this gap, the D-TASK protocol, which consists of a secret generation primitive and a three-round consensus, is proposed. To our knowledge, this is the first protocol to fulfill all three properties for deploying shared information across a TEE cluster.

2. TEAR-DO mechanism. The TEAR-DO mechanism comprises a subproblem assignment scheme and a Byzantine node detection scheme. The former scheme allocates each subproblem to only several (e.g., 4) designated TEEs, which significantly reduces redundant computations on each subproblem. Instead of using remote attestations, the latter scheme achieves that every honest TEE can identify Byzantine nodes by directly verifying the validity of signatures. Since generating and verifying a signature only consumes around 2 milliseconds, the second challenge caused by the time-inefficient remote attestations is then resolved. Moreover, supported by the former scheme and the unique design by enabling signatures to detect Byzantine nodes, the third challenge is also resolved.

3. As a supplementary contribution of the paper, theoretical analysis proves that the framework can preserve data privacy and defend against Byzantine attacks when solving an energy market clearing optimisation problem. Comprehensive numerical simulations are conducted to evaluate the performance of the framework by comparing it with the state-of-the-art technique [241]. The results yield three key results: (i) in attack-free scenarios, the framework reduces the P2P market clearing time by up to a factor of ten compared to the method [241]; (ii) the framework neutralises side effects of Byzantine attacks by introducing an average of only three redundant computations per subproblem; (iii) the framework allows for the integration with second-order DO methods, enabling real-time P2P market clearing even under 10,000 trading pairs. These findings collectively confirm that the proposed framework provides a practical solution for real-world P2P energy markets.

The remainder of this paper is organised as follows. Section 6.2 formats the

optimisation problem of the P2P energy markets. Section 6.3 introduces the preliminaries and the framework for the P2P energy market clearing process. Sections 6.4 and 6.5 present the technical details of the proposed protocol and mechanism, respectively. Section 6.6 presents the theoretical analysis on the framework. Section 6.7 reports and discusses the simulation results. Section 6.8 concludes the paper.

6.2 Problem Formation

This section first introduces the optimisation problems [237, 241] widely formulated in P2P energy trading, followed by the description of their corresponding DO algorithms.

6.2.1 Optimisation Problem

In a real-time P2P energy market, a set of prosumers engage in P2P energy trading, while a grid operator oversees the process to ensure grid integrity. Within each market clearing interval, a prosumer's role is either a buyer or a seller. The total numbers of buyers and sellers in the interval are denoted by *M* and *N*, respectively. The objective of P2P energy market clearing (denoted as F) is to maximise the social welfare of the participant population while satisfying the network constraints of the underlying power system:

$$\max F = \sum_{m\in[M]} U_m(e_m) - \sum_{n\in[N]} C_n(e_n) \tag{6.1}$$

The notation "$[*]$" is short for $\{1,\dots,*\}$ and $*\geq 2$ is an arbitrary integer; $U_m(e_m)$ $(\forall m \in [M])$ is a quadratic utility function representing the utility of buyer *m* from purchasing e_m units of energy; $C_n(e_n)$ $(\forall \mathrm{n} \in [N])$ is a quadratic cost function representing the cost of seller *n* for selling e_n units of energy.

Two types of network topology (i.e., meshed network and radial network) that are widely deployed in current power systems are considered. Characterised by interconnected loops, meshed networks ensure high reliability for urban centres, enabling dynamic, market-driven P2P energy trading models with multidirectional

energy flows [252]. Under certain assumptions [232], meshed network-based P2P energy markets are often modelled through power transfer distribution factors (PTDFs), and the only explicit physical constraint in the PTDF-based meshed network is the thermal limit:

$$-F_l^{max} \le F_l \le F_l^{max}, \forall l \in [L] \tag{6.2}$$

where L is the number of power delivery lines in the meshed network. F_l is the power flow in the l-th line and F_l^{max} is its maximum transmitted power limit of the line.

In contrast, as a low-cost, tree-like topology, radial networks suit localised P2P markets [237]. The linearised DistFlow model is widely applied to radial networks [253]. Under this combination, the physical constraints in the DistFlow model-based radial networks are described in Eqs. (6.3a)-(6.3d):

$$P_i^{\text{bus}} = \sum_{(i,j)\in[L]} P_{ij}\,, Q_i^{\text{bus}} = \sum_{(i,j)\in[L]} Q_{ij}\,, \forall i \in [\mathcal{V}] \tag{6.3a}$$

$$LD_{ij}^{v} = v_i - v_j - 2\left(r_{ij}P_{ij} + x_{ij}Q_{ij}\right) = 0, \forall (i,j) \in [L] \tag{6.3b}$$

$$P_{ij}^2 + Q_{ij}^2 \le \left(S_{ij}^{\max}\right)^2, \forall (i,j) \in [L] \tag{6.3c}$$

$$\left(V_i^{\min}\right)^2 \le v_i \le (V_i^{\max})^2, \forall i \in [I] \tag{6.3d}$$

Eq. (6.3a) denotes nodal power balance constraint. The sum of active (reactive) branch power flows leaving bus i equals to bus active (reactive) power injections P_i^{bus} (Q_i^{bus}) of bus i. P_{ij} (Q_{ij}) is the active (reactive) branch power flow from bus i to bus j. Eq. (6.3b) represents the voltage drop constraint based on the linearised DistFlow model, where v_i is the squared voltage magnitude at bus i, and (r_{ij}, x_{ij}) is the impedance from bus i to bus j. Eq. (6.3c) denotes thermal limit constraint and the power on each branch must not exceed its rating $S_{ij}^{\max}$. Eq. (6.3d) denotes voltage-magnitude limit constraint and v_i must stay within statutory bounds defined by the minimum allowable voltage magnitude $V_i^{\min}$ and the maximum allowable voltage magnitude $V_i^{\max}$.

Besides the network-layer constraints, the optimisation problem in Eq. (6.1) is also

subject the market-layer constraints:

$$e_m^{min} \leq e_m = \sum_{n \in [N]} e_{mn} \leq e_m^{max}, \forall m \in [M] \tag{6.4a}$$

$$e_n^{min} \leq e_n = \sum_{m \in [M]} e_{nm} \leq e_n^{max}, \forall n \in [N] \tag{6.4b}$$

$$e_{mn} = e_{nm} \tag{6.4c}$$

In Eq. (6.4a), the total units of energy e_m purchased by buyer m must be within the minimum (e_m^{min}) and maximum (e_m^{max}) energy purchasing units. In Eq. (6.4b), the total units of energy e_n sold by seller n must be within the minimum (e_n^{min}) and maximum (e_n^{max}) energy selling units. Eq. (6.4c) denotes coupling constraints. e_{mn} are the units of energy purchased by buyer *m* from seller *n*, and e_{nm} are the units of energy sold by seller *n* to buyer *m*.

6.2.2 Distributed Optimisation Algorithms

Under meshed networks, the problem in Eq. (6.1) subject to Eq. (6.2) and Eqs. (6.4a)-(6.4c) is solved by the DO algorithm (see Eqs. (20)-(41) in [232]). Under radial networks, the problem in Eq. (6.1) subject to Eqs. (6.3a)-(6.3d) and Eqs. (6.4a)-(6.4c) can be solved by the ALADIN algorithm [254]. Currently, since no research uses the ALADIN algorithm to solve the optimisation problem in Eq. (6.1), this paper presents the workflows of the ALADIN algorithm as follows:

In an iteration r, $r = 0, 1, ...$, the ALADIN algorithm divides the problem in Eq. (6.1) into three blocks. The first block defines a total of *M*+*N* optimisation subproblems, each corresponding to a prosumer. The second block defines a feasibility subproblem subject to Eqs. (6.3a)-(6.3d). The *M*+*N*+1 subproblems are executed in parallel. Apart from the two blocks, a coordinating block, that collects the updated results of the above two blocks, is defined to update coordinating global variables. Specifically, two steps are sequentially executed in each iteration:

Step 1 – A optimisation subproblem associated with a prosumer ℓ ($\ell \in [M] \cup [N]$)

at bus i ($i \in [I]$) subject to Eq. (6.4a) or Eq. (6.4b) is solved to update a decision vector $\boldsymbol{u}_\ell$:

$$\boldsymbol{u}_\ell = [\boldsymbol{e}_\ell{}^\top, P_{\ell,i}^{\text{in}}]^\top, \boldsymbol{e}_\ell = [e_{\ell 1}, \dots, e_{\ell K}]^\top, P_{\ell,i}^{\text{in}} = \frac{\sigma_\ell}{\Delta t} \sum_{k \in [K]} e_{\ell k} \tag{6.5a}$$

$$\begin{aligned} \boldsymbol{u}_\ell^{(r)} \leftarrow \arg\min \{ & f_\ell(\boldsymbol{e}_\ell) + (\boldsymbol{\lambda}_\ell^{\text{eq},(r-1)})^\top \boldsymbol{e}_\ell \\ & + \frac{\sigma}{\Delta t} \left(\boldsymbol{\lambda}^{\text{b},(r-1)}\right)^\top \boldsymbol{e}_\ell + \frac{\rho^{\text{eq}}}{2} ||\boldsymbol{e}_\ell - \boldsymbol{z}^{\text{eq},(r-1)}||_2^2 \end{aligned} \tag{6.5b}$$

where $\boldsymbol{e}_\ell$ (when $\ell \in [M]$, $K = N$, when $\ell \in [N]$, $K = M$) records K bilateral trading variables, $P_{\ell,i}^{\text{in}}$ is the power injection of prosumer ℓ at bus i, $\sigma_\ell = -1$ when $\ell \in [M]$ and $\sigma = 1$ when $\ell \in [N]$, $f_\ell(\boldsymbol{e}_\ell)$ is the local objective function ($-U_m$ or C_n in (1), $\boldsymbol{\lambda}_\ell^{\text{eq},(r-1)}$ and $\boldsymbol{z}^{\text{eq},(r-1)}$ are the market-based coordinating variable vectors, $\lambda_\ell^{\text{b},(r-1)}$ is the bus-based dual variable, and ρ^{eq} is a positive penalty parameter. After that, the first order vector $\boldsymbol{g}_\ell^{(r)}$ of Eq. (6.5b) with respect to $\boldsymbol{e}_\ell$ is computed. $\boldsymbol{u}_\ell^{(r)}$, $\boldsymbol{g}_\ell^{(r)}$ along with the coefficients of U_m or C_n are sent to a master.

Concurrently, a feasibility subproblem is defined to update a decision vector $\boldsymbol{P}^{\text{b}}$ subject to Eqs. (6.3a)-(6.3d):

$$\boldsymbol{P}^{\text{b},(r)} \leftarrow \arg\min -\left(\boldsymbol{\lambda}^{\text{b},(r-1)}\right)^\top \boldsymbol{P}^{\text{b}} + \frac{\rho^{\text{b}}}{2} ||\boldsymbol{P}^{\text{b}} - \boldsymbol{z}^{\text{b},(r-1)}||_2^2 \tag{6.6}$$

where $\boldsymbol{P}^{\text{b},(r)}$ records I active power injection values, $\boldsymbol{\lambda}^{\text{b},(r-1)}$ contains I bus-based dual variables regarding active power injections, $\rho^{\text{b}} > 0$ is a penalty parameter. After that, the first order vector $\boldsymbol{g}^{\text{b},(r)}$ of Eq. (6.6) with respect to $\boldsymbol{P}^{\text{b}}$ is computed. $\boldsymbol{P}^{\text{b},(r)}$ and $\boldsymbol{g}^{\text{b},(r)}$ are then sent to the master.

Step 2– The master collects all updated coupling variables and first order vectors to compute the second-order Hessian matrix $\boldsymbol{H}$ [254]. It then updates the coordinating vectors $\boldsymbol{z}^{(r)}$ and $\boldsymbol{\lambda}^{(r)}$ by solving the global optimisation subproblem:

$$\{\boldsymbol{z}^{(r)}, \boldsymbol{\lambda}^{(r)}\} \leftarrow \arg\min \boldsymbol{g}^{(r)} \Delta \boldsymbol{z} + \Delta \boldsymbol{z}^\top \boldsymbol{H} \Delta \boldsymbol{z}$$

$$\text{s.t.} \, \boldsymbol{P}^{\text{in}} - \boldsymbol{P}^{\text{b}} = 0, e_{mn} - e_{nm} = 0, \forall m \in [M], \forall n \in [N] \tag{6.7a}$$

$$\boldsymbol{\lambda}^{(r)} = \left[\boldsymbol{\lambda}_1^{\text{eq},(r)}, \dots, \boldsymbol{\lambda}_\ell^{\text{eq},(r)}, \dots, \boldsymbol{\lambda}_{M+N}^{\text{eq},(r)}, \boldsymbol{\lambda}^{\text{b},(r)}\right] \tag{6.7b}$$

$$\boldsymbol{z}^{(r)} = \left[\boldsymbol{z}_1^{\mathrm{eq},(r)}, \ldots, \boldsymbol{z}_\ell^{\mathrm{eq},(r)}, \ldots, \boldsymbol{z}_{M+N}^{\mathrm{eq},(r)}, \boldsymbol{z}^{\mathrm{b},(r)}\right] \tag{6.7c}$$

where $\Delta\boldsymbol{z} = \boldsymbol{z} - \boldsymbol{u}^{(r)}$, $\boldsymbol{u}^{(r)} = [\boldsymbol{u}_1^{(r)}, \ldots, \boldsymbol{u}_\ell^{(r)}, \ldots, \boldsymbol{u}_{M+N}^{(r)}, \boldsymbol{P}^{\mathrm{b},(r)}]$, $\boldsymbol{g}^{(r)} = [\boldsymbol{g}_1^{(r)}, \ldots, \boldsymbol{g}_{M+N}^{(r)}, \boldsymbol{g}^{\mathrm{b},(r)}]$, $\boldsymbol{P}^{\mathrm{b}} = [P_1^{\mathrm{b}}, \ldots, P_i^{\mathrm{b}} \ldots, P_I^{\mathrm{b}}]$ and $\boldsymbol{P}^{\mathrm{in}} = [P_1^{\mathrm{in}}, \ldots, P_i^{\mathrm{in}}, \ldots, P_I^{\mathrm{in}}]$, $P_i^{\mathrm{in}} = \sum_{\ell \in [M_i] \cup [N_i]} \sum_{k \in [K]} \frac{\sigma_\ell}{\Delta t} e_{\ell k}$, M_i and N_i are the number of buyers and sellers at bus i, respectively. In the next section, the framework used to execute the DO algorithms will be elaborated.

6.3 Preliminaries and the Framework of the P2P Energy Market Clearing

Before introducing the technical implementation details of the proposed framework, in this section the preliminaries of the framework are first presented, followed by the description of the framework designed for the P2P energy market clearing.

6.3.1 Preliminaries

BC and TEEs are considered as the building blocks of the proposed framework for the P2P energy market clearing. The details of BC and TEEs are presented as follows.

Blockchain. BC is a decentralised platform comprising B nodes, with the b-th node in the BC indexed as N_b $(b \in [B])$. To make the BC cyberattack-resilient, it must meet two conditions: *Condition 1* – For a set of $3f + 1$ nodes, there are up to f Byzantine nodes that are fully controlled by an adversary. *Condition 2* – it operates under a "Δ-synchronous condition" [255]; that is, any message sent at time t is guaranteed to be received by the receiver by time $t + \Delta$ (where the symbol "Δ" denotes a time duration). Instead of using smart contracts to provide trustworthy computation via redundancy, this paper first leverages the BC's consensus protocol [255] to ensure that all BC nodes maintain an identical set of data used for each energy market clearing interval. Then, the lightweight (TEAR)-DO mechanism is proposed to

ensure the computational integrity during executing DO algorithms.

Trusted *execution environment*. Each BC node is equipped with a TEE, which is a hardware-protected area within a processor [250]. A TEE ensures that the data stored in it is only known by itself. Each TEE TEE_b ($b \in [B]$) naturally configures with two unique key pairs (sks_b, pks_b) and (skd_b, pke_b), where sks_b and skd_b is privately stored in TEE_b; (sks_b, pks_b) is used for generation and verification of signatures [256]; and (skd_b, pke_b) is used for message encryption and decryption. TEEs serve a dual, critical role: they are central to preserving the privacy of sensitive energy data (e.g., prosumers' bids and offers, and the grid's network parameters), while also acting as a building block in the TEAR-DO mechanism to execute DO algorithms and detect Byzantine attacks.

6.3.2 Framework of the P2P Energy Market Clearing

This paper introduces a three-phase P2P energy market clearing framework, as shown in Figure 6-1. In this framework, the computational tasks of a DO algorithm are offloaded from prosumers and executed securely within TEEs:

In the first phase, prosumers send their encrypted energy trading request to a BC. The request of a prosumer includes: (i) the index of the bus where the prosumer is located; (ii) the minimum and maximum amounts of energy the prosumer intends to purchase or sell; and (iii) the coefficients of its utility or cost function. In the meantime, the grid operator sends an encrypted request containing the grid parameters to the BC.

In the second phase, the request allocation period ($\mathcal{T}_1$) is dedicated to collecting participants' requests. During this interval, the BC executes its consensus protocol to commit all valid, encrypted requests into blocks. Once a new block is finalised, every BC node sends the block to its controlled TEE.

The third phase is an energy market clearing phase ($\mathcal{T}_2$), wherein the TEEs collectively execute DO algorithms using the data contained within the blocks generated during $\mathcal{T}_1$.

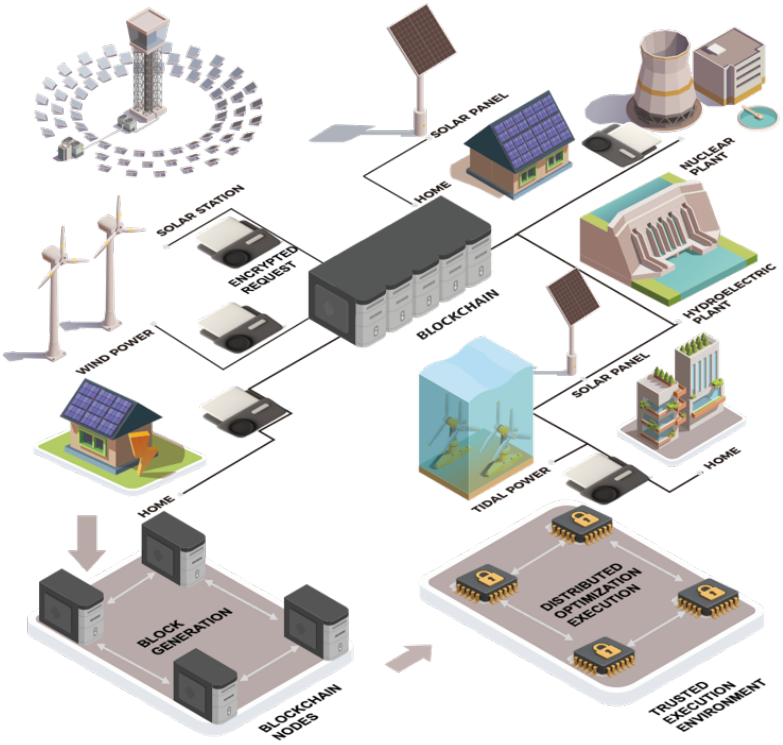

FIGURE 6-1. The three-phase framework of P2P energy market clearing.

6.3.3 Grid-operator regulatory interface before BC admission.

In a practical deployment, the first phase can include a grid-operator pre-admission check before a prosumer's encrypted energy trading request becomes valid for a market clearing interval. The grid operator uses the current physical operating state to decide whether a prosumer's request may enter that interval, and this decision is made before the request is committed by the BC. The check uses only the fields needed for distribution-network constraint assessment, such as the prosumer's bus index and requested maximum purchasing or selling amount. The utility or cost-function coefficients remain encrypted for the later TEE-based DO computation. If the admitted requests in an interval may overload a line or violate network constraints, the grid operator can publish interval-specific admission rules. These rules may refer to Eq. (6.2), Eqs. (6.3a)-(6.3d), bus locations, forecasted congestion and available network capacity. Prosumers satisfying the rules may submit encrypted requests to the BC, while other candidate requests are rejected for the current interval or deferred to a later

interval. This design is consistent with network-constrained P2P energy trading studies and dynamic operating envelope studies. A dynamic operating envelope is an interval-specific import or export limit assigned by the network operator to help enforce distribution-grid constraints during market clearing [233, 241, 297].

This grid-operator action is therefore an admission interface rather than an intervention inside the TEAR-DO computation. Once the admitted requests and encrypted grid parameters are committed to the BC, all selected TEEs execute the same DO algorithm over the same recorded data. The grid operator determines the valid request set and submits the corresponding grid parameters before clearing; TEAR-DO then addresses privacy preservation and Byzantine-tolerant DO after admission. To make this pre-admission check auditable, the BC can record the admission rules and admitted-request identifiers for each interval. This preserves decentralised execution after admission while retaining a clear regulatory interface for distribution-network constraints. Fairness policies for deferred prosumers and emergency-restoration procedures are outside this chapter and left to future work.

6.3.4 Design challenges and goals.

As established in Section 6.1.2, the framework must address the three key challenges to solve the identified trilemma in P2P energy markets.

The first challenge is to establish the foundation for an efficient decryption scheme, which hinges on the secure deployment of the secret key *SKS* and the script $\mathbb{C}$ to all TEEs. This deployment must satisfy the following three properties:

1) **Strict Confidentiality:** The secret key *SKS* must be known exclusively to the selected TEEs.

2) **Full Decentralisation:** No 3rd party can be relied on during the deployment process.

3) **Attack resilience:** 1) and 2) can be achieved even in the presence of Byzantine nodes who can perform any possible misbehaviours to their controlled TEEs.

The second challenge is the need to replace the slow remote attestations with an efficient proof scheme capable of verifying computational correctness of TEEs' outputs.

The third challenge is ensuring that the DO algorithm achieves optimal convergence with minimal computational redundancy, even in the presence of Byzantine nodes.

This paper resolves the three challenges in sequence: Section 6.4 introduces the D-TASK protocol for secure key and script deployment, thereby addressing the first challenge, while Section 6.5 presents the TEAR-DO mechanism including a task assignment scheme and Byzantine-tolerant DO scheme, which addresses the second and third challenges.

6.4 D-Task Protocol

The decentralised TEE attested secret key deployment (D-TASK) protocol selects TEEs that verifiably deploy the pre-defined script and the secret key SKS. As shown in Figure 6-2, D-TASK consists of a secret generation primitive and a three-round consensus. It is executed as a setup protocol before repeated online market clearing intervals, rather than during every interval. Therefore, D-TASK is an offline protocol and not the online bottleneck when the number of prosumers grows.

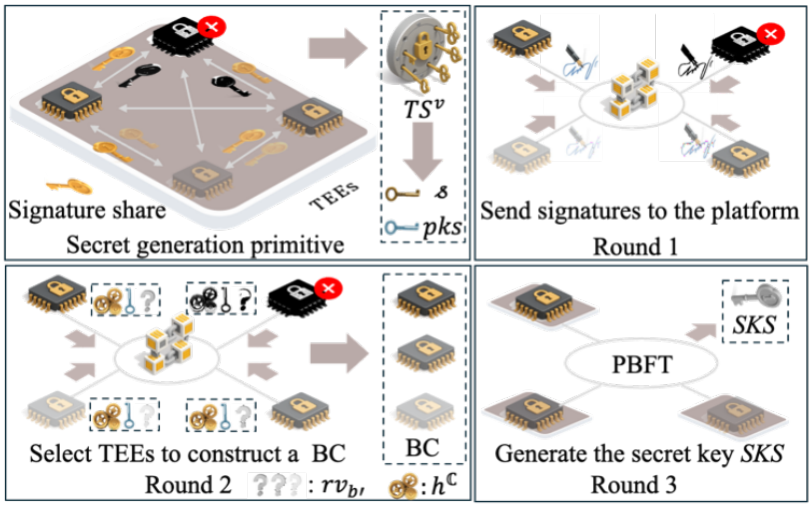

FIGURE 6-2. The workflows of the D-TASK protocol.

6.4.1 Secret Generation Primitive

The proposed secret generation primitive defines an one-round P2P communication process which ensures that a group of TEEs generate a same secret $\mathcal{s}$ and only these TEEs know the secret $\mathcal{s}$. Given $B' = 3f' + 1$ TEEs with up to f' TEEs hosted by Byzantine nodes, the process of the primitive is depicted from the perspective of the TEE $TEE_{b'}$ $(\forall b' \in [B'])$:

At time point t, $TEE_{b'}$ signs a value v using its own secret key $sks_{b'}$, which outputs a share (denoted as $S_{b'}^{v}$):

$$S_{b'}^{v} = Ss(v, sks_{b'}) \tag{6.8}$$

where $Ss(\cdot)$ represents the share signing algorithm. $TEE_{b'}$ then encrypts $S_{b'}^{v}$ using $TEE_{a'}$'s public key $pke_{a'}$:

$$E_{a'}(S_{b'}^{v}) = Enc(S_{b'}^{v}, pke_{a'}), \mathrm{a}' \in [B'] \tag{6.9}$$

where $Enc(\cdot)$ represents the encryption algorithm; $E_{a'}(S_{b'}^{v})$ is the encrypted version of $S_{b'}^{v}$ with respect to $TEE_{a'}$. $TEE_{b'}$ outputs $E_{a'}(S_{b'}^{v})$ to its host node $N_{b'}$, and $N_{b'}$ further sends $E_{a'}(S_{b'}^{v})$ to node $N_{a'}$. $N_{a'}$ then inputs $E_{a'}(S_{b'}^{v})$ to its $TEE_{a'}$.

After receiving $E_{b'}(S_{a'}^{v})$ from $TEE_{a'}$, $TEE_{b'}$ decrypts $E_{b'}(S_{a'}^{v})$ through $skd_{b'}$ and a decryption algorithm $Dec(\cdot)$:

$$S_{a'}^{v} = Dec(E_{b'}(S_{a'}^{v}), skd_{b'}) \tag{6.10}$$

$TEE_{b'}$ then checks the validity of $S_{a'}^{v}$ using $TEE_{a'}$'s public key $pks_{a'}$ and a verification algorithm $Sv(\cdot)$:

$$Sv(S_{a'}^{v}, pks_{a'}) \rightarrow \{0,1\} \tag{6.11}$$

If Eq. (6.11) outputs 1, it stores $S_{a'}^{v}$; otherwise, it deletes $S_{a'}^{v}$.

At $t + \Delta$, upon storing $2f' + 1$ shares, $TEE_{b'}$ generates a threshold signature (denoted as TS^{v}) and a secret $\mathcal{s}$:

$$TS^{v} = TSg(\dots, S_{b'}^{v}, \dots, S_{a'}^{v}, \dots) \tag{6.12}$$

$$\mathcal{s} = H(TS^{v}) \tag{6.13}$$

where $TSg(\cdot)$ is the threshold signature generation algorithm and $H(\cdot)$ is a hash function.

Since f' Byzantine nodes could prevent their TEEs from generating $\mathcal{s}$ by any possible attacks, an additional process is required to eliminate these TEEs failing to generate $\mathcal{s}$. Then, the D-TASK protocol introduces a three-round consensus.

6.4.2 Three-Round Consensus

The three-round consensus is a decentralised protocol that can be built upon any decentralised platforms satisfying the two conditions defined in Section 6.3.1. The three-round consensus selects a final set of B TEEs that have verifiably deployed the same $\mathcal{s}$ and $\mathbb{C}$. Subsequently, the B TEEs privately generate the secret key *SKS* and corresponding public keys.

Before the start of the consensus, each TEE first generates a public key $pks = g^{\mathcal{s}}$, where g is a generator of a cyclic group. Each honest node then deploys the pre-specified script $\mathbb{C}$ to its TEE. After that, each TEE generates a hash value $h^{\mathbb{C}} = H(\mathbb{C})$.

From the perspective of $TEE_{b'}$ $(\forall b' \in [B'])$, once the public key pks and the hash value $h^{\mathbb{C}}$ are generated, the three-round consensus begins. At the first round, $TEE_{b'}$ signs $h^{\mathbb{C}}$ to generate a signature $Sig_{b'}$ using $\mathcal{s}$, a signing algorithm $Sign(\cdot)$, and a random value $rv_{b'}$, where $rv_{b'}$ is only known to $TEE_{b'}$ itself.

$$Sig_{b'} = Sign((h^{\mathbb{C}}, rv_{b'}), \mathcal{s}) \tag{6.14}$$

$TEE_{b'}$ outputs $Sig_{b'}$ to its host node $N_{b'}$, and $N_{b'}$ sends $Sig_{b'}$ to a decentralised platform at time $t + 2\Delta$. Consider there are B'' encrypted messages stored in the platform at $t + 3\Delta$, the missing $B' - B''$ nodes are firstly eliminated.

At the second round, $TEE_{b'}$ outputs $rv_{b'}$, $h^{\mathbb{C}}$ and pks to $N_{b'}$ which then sends them to the decentralised platform at $t + 3\Delta$. Algorithm $V_c(\cdot)$ is applied to check the validity of $Sig_{b'}$:

$$V_c(Sig_{b'}, rv_{b'}, h^{\mathbb{C}}, pks) \rightarrow \{0,1\} \tag{6.15}$$

If Eq. (6.15) outputs "1", $N_{b'}$, together with its $TEE_{b'}$, is selected as a BC node; otherwise, $N_{b'}$ is eliminated. Therefore, total B nodes, whose signatures are validated through Eq. (6.15) before $t + 4\Delta$, are selected as the BC nodes ($B \leq B'' \leq$

B').

At the third round, a Byzantine fault tolerance consensus process is initiated by a selected leader TEE (e.g., TEE_b, $b \in [B]$). TEE_b generates a certificate $\mathfrak{C}$, encrypts it through Eq. (6.9), and sends the encrypted $\mathfrak{C}$ (denoted as $E_a(\mathfrak{C})$) to TEE_a for every $a \in [B]$. After the consensus, TEE_a decrypts $E_a(\mathfrak{C})$ through Eq. (6.10) and generates *SKS* through Eq. (6.16):

$$SKS = H(s \bmod \mathfrak{C}), \text{PKE} = G_E(SKS), PVK = G_V(SKS) \tag{6.16}$$

where $G_E(\cdot)$ and $G_V(\cdot)$ are the key generation functions for the encryption key *PKE* and the verifying key *PVK*, respectively.

6.4.3 Properties analysis

In the secret generation primitive, its strict confidentiality property is achieved by the encryption and decryption operations (see Eq. (6.9) and Eq. (6.10)) and the privacy nature of TEEs. Its fully decentralisation property is achieved since the generation of s relies on a one-round P2P share exchange process and no 3rd party involves in the process. As for its attack resilience property, since there are at most f' Byzantine nodes among the $3f'+1$ and only $2f'+1$ valid shares are needed to generate the secret s, every TEE can receive at least $2f'+1$ valid shares from the $2f'+1$ honest nodes even if the f' Byzantine nodes may perform any possible misbehaviours.

In the three-round consensus, its strict confidentiality property is achieved in the third round - only the B TEEs can access the certificate $\mathfrak{C}$ and generate *SKS*. Its full decentralisation property is achieved, as the first two rounds rely on a decentralised platform and the third round relies on a Byzantine fault tolerance consensus process. As for its attack resilience property, in the first two rounds, the signature verification algorithm of Eq. (6.15) and the Δ-synchrony defined in condition 2 eliminates the nodes that fail to provide a valid signature to attest their deployments on s and $\mathbb{C}$. In the third round, the Byzantine fault tolerance consensus process is sufficient to defend against any Byzantine behaviours.

6.5 TEAR-DO Mechanism

The time efficient and attack resistant (TEAR)-DO mechanism employs two components to ensure the time-efficient and attack-resilient execution of DO algorithms. The first component allocates each subproblem to κ TEEs to limit redundant computation. However, this approach creates a critical vulnerability: if the κ TEEs assigned to a subproblem are all controlled by Byzantine nodes, the subproblem cannot be correctly executed. Then, the second component, a proof-based detection scheme, is designed to resolve the issue that a subproblem is only assigned to Byzantine nodes.

6.5.1 Task Assignment Scheme

The scheme is designed to randomly assign a subproblem to κ TEEs. Every BC node inputs an identical random number rn into its TEE. For a subproblem $\mathcal{SP}$ defined in a DO algorithm, TEE_b ($b \in [B]$) uses the algorithm $Assign(\cdot)$ to determine whether it should solve the subproblem $\mathcal{SP}$:

$$\boldsymbol{S}_{\mathcal{SP}} = Assign(rn, \mathcal{SP}, \kappa) = \underset{|\boldsymbol{S}_{\mathcal{SP}}|=\kappa}{\text{Arg min}}\{h_1, \dots, h_B\} \tag{6.17}$$

$$h_b = H(rn|\mathcal{SP}|b) \tag{6.18}$$

where "|" denotes the concatenation operation and h_b is the hash value associated with TEE_b. $Assign(\cdot)$ selects the first κ minimum hash values from $\{h_1, \dots, h_B\}$ and inputs them into the set $\boldsymbol{S}_{\mathcal{SP}}$. If h_b is in $\boldsymbol{S}_{\mathcal{SP}}$, TEE_b executes the subproblem $\mathcal{SP}$; otherwise, it does not execute it. This scheme is executed once a block is generated in Phase 2 (see Section 6.3.2).

6.5.2 Byzantine Node Detection Scheme

The scheme fully uses the secret key *SKS* and the script $\mathbb{C}$ deployed in each selected TEE to detect Byzantine nodes. This detection can be guaranteed due to the observation – a signature-based message validation strategy can be further extended to detect incorrect computational results of TEEs through the three facts that are achieved in the

proposed framework:

First, except for the B TEEs, no one can use the secret key SKS to generate a valid signature.

Second, a validation rule (i.e., a code sniff within $\mathbb{C}$) mandates a specific action: upon completing any computation, the TEE must immediately generate a signature using *SKS*. This signature covers a hash of the entire computational context, including the inputs, the operational codes, and the resulting outputs regarding the computation.

Third, for any given subproblem (e.g., the equations of Eq. (6.2)), its components – the operational codes and all required inputs – are unambiguously defined by $\mathbb{C}$. Moreover, $\mathbb{C}$ contains the operational codes and static inputs (e.g., constant parameters), while dynamic inputs (e.g., sellers' offers) are stored in the blocks of the BC.

Based on the second and the third facts, when a TEE generates a subproblem's output, the output is uniquely bound to these exact components. Supported by the TEE's computational integrity – it can faithfully report what data and codes are used to generate an output, if a TEE deviates from the specified computation by using incorrect codes, fetching invalid inputs, forging an output or generating a signature without using SKS, the signature verification process designed in the proposed Byzantine node detection scheme will effectively identify these misbehaviours. In detail, the scheme has the two processes and a countermeasure:

1) Proof generation process

In an arbitrary iteration r of a DO process, TEE_b ($b \in [B]$) executes the assigned subproblems to generate a set $\boldsymbol{\mathcal{S}}_b^{(r)}$ which contains many updated variables. TEE_b encrypts $\boldsymbol{\mathcal{S}}_b^{(r)}$ (denoted as $(\boldsymbol{\mathcal{S}}_b^{(r)})^E$) using *PKE*, and generates a proof $p_b^{(r)}$ using the signing algorithm $Sg_s(\cdot)$ and SKS:

$$p_b^{(r)} = Sg_s\left(H\left((\boldsymbol{\mathcal{S}}_b^{(r)})^E|\boldsymbol{O}^{do}|\boldsymbol{D}^{do}|\boldsymbol{In}^{do}|\mathbb{C}\right), b, SKS\right) \tag{6.19}$$

$$(\boldsymbol{\mathcal{S}}_b^{(r)})^E = Enc\left(\boldsymbol{\mathcal{S}}_b^{(r)}, PKE\right) \tag{6.20}$$

where $Enc(\cdot)$ is the encryption algorithm, $\boldsymbol{O}^{do} \in \mathbb{C}$ is the set of operational codes regarding the assigned subproblems, and the inputs of $\boldsymbol{O}^{do}$ consist of the constant parameters $\boldsymbol{D}^{do} \in \mathbb{C}$ and the dynamic variables $\boldsymbol{In}^{do}$ (e.g., buyers bids or previous iteration's outputs) from current and previous iterations. $p_b^{(r)}$ is used to attest that $\boldsymbol{S}_b^{(r)}$ is generated within TEE_b by executing $\boldsymbol{O}^{do}$ with the inputs $\boldsymbol{D}^{do}$ and $\boldsymbol{In}^{do}$ at the r-th iteration. TEE_b sends $\{\left(\boldsymbol{S}_b^{(r)}\right)^E, H(\boldsymbol{O}^{do}|\boldsymbol{D}^{do}|\boldsymbol{In}^{do}), b, p_b^{(r)}\}$ to its host node N_b, and N_b broadcasts it all other TEEs.

2) Byzantine Node Detection Process

When a TEE (e.g., TEE_a, $a \neq b$) receives $\{(\boldsymbol{S}_b^{(r)})^E, H(\boldsymbol{O}^{do}|\boldsymbol{D}^{do}|\boldsymbol{In}^{do}), b, p_b^{(r)}\}$, it determines the operations $\boldsymbol{O'}^{do}$, the parameters $\boldsymbol{D'}^{do}$ and the variables $\boldsymbol{In'}^{do}$ based on the script $\mathbb{C}$ and the involved subproblems. TEE_a then checks: (i) whether $H(\boldsymbol{O}^{do}|\boldsymbol{D}^{do}|\boldsymbol{In}^{do})$ is equal to $H\left(\boldsymbol{O'}^{do}|\boldsymbol{D'}^{do}|\boldsymbol{In'}^{do}\right)$, and (ii) whether $p_b^{(r)}$ is valid:

$$V_s\left(p_b^{(r)}, H\left((\boldsymbol{S}_b^{(r)})^E|\boldsymbol{O'}^{do}|\boldsymbol{D'}^{do}|\boldsymbol{In'}^{do}|\mathbb{C}\right), b, PVK\right) \rightarrow \{0,1\} \tag{6.21}$$

where $V_s(\cdot)$ is the signature verification algorithm. If $H(\boldsymbol{O}^{do}|\boldsymbol{D}^{do}|\boldsymbol{In}^{do}) \neq H\left(\boldsymbol{O'}^{do}|\boldsymbol{D'}^{do}|\boldsymbol{In'}^{do}\right)$ and/or $V_s(\cdot) \neq 1$, TEE_a identifies N_b as a Byzantine node. Once $p_b^{(r)}$ is validated, TEE_a decrypts $(\boldsymbol{S}_b^{(r)})^E$ through the decryption algorithm $Dec(\cdot)$ and the secret key SKS:

$$\boldsymbol{S}_b^{(r)} = Dec\left((\boldsymbol{S}_b^{(r)})^E, SKS\right) \tag{6.22}$$

Notably, the scheme enforces that every TEE needs to broadcast its outputs to all other TEEs in each iteration for security purposes. By leveraging the adopted Δ-synchronous condition (see Section 6.3.1), if TEE_a broadcasts $(\boldsymbol{S}_a^{(r)})^E$ at time point t but does not receive a valid $(\boldsymbol{S}_b^{(r)})^E$ by $t + \Delta$, TEE_a also identifies N_b as a Byzantine node.

Countermeasure: The countermeasure is designed to ensure optimal convergence of DO when Byzantine attacks are performed: if TEE_a does not receive a valid

$(\boldsymbol{S}_b^{(r)})^E$ at the end of $t + \Delta$, it computes the missing variables that are supposed to be in $\boldsymbol{S}_b^{(r)}$ by itself in the current and the following iterations. This design ensures that even if a subproblem is assigned to κ Byzantine nodes, the subproblem will still be correctly executed in each iteration. Intuitionally, the countermeasure may increase the redundant computing. However, our simulations (Figure 6-5) show that a κ value of 5 limits the redundant computations for any subproblem to 4 or fewer with a high probability.

6.6 Theoretical Analysis

This section analyses the privacy preserving and Byzantine tolerant properties of the framework, while the time-efficiency of the framework is evaluated in Section 6.7.

Regarding the data privacy property, the sensitive data in each request is first encrypted using the *PKE*. Supported by the D-TASK protocol, only the *B* selected TEEs who have *SKS* can decrypt the encrypted requests. Since no one can access the data stored in TEEs, the privacy of participants' sensitive data is preserved in the framework.

Regarding the Byzantine tolerance property, a successful Byzantine attack is one where an incorrect output $\boldsymbol{S'}_b^{(r)} \neq \boldsymbol{S}_b^{(r)}$ passes the verification in Eq. (6.21). Byzantine nodes could achieve this via two primary paths: (i) extracting the secret key *SKS* to forge signatures, or (ii) compelling a TEE to sign arbitrary, malicious contents.

Path (i) is rendered infeasible by the D-TASK protocol and TEE's hardware protections. The D-TASK protocol ensures only the *B* selected TEEs knows the secret key *SKS*, while TEEs prevent the extraction of the secret key *SKS*.

Path (ii) is blocked by the predefined validation rule in $\mathbb{C}$ and the adoption of the BC. The rule in $\mathbb{C}$ mandates that every TEE can only sign a hash of the entire computational context — including the specific inputs, the operations performed, the encrypted outputs of the operations, and $\mathbb{C}$, i.e., $H((\boldsymbol{S}_b^{(r)})^E|\boldsymbol{O}^{do}|\boldsymbol{D}^{do}|\boldsymbol{In}^{do}|\mathbb{C})$ in Eq. (6.20). If Byzantine nodes change any elements in the hash, the honest verifier will find

$H(\boldsymbol{O}^{do}|\boldsymbol{D}^{do}|\boldsymbol{In}^{do}) \neq H(\boldsymbol{O'}^{do}|\boldsymbol{D'}^{do}|\boldsymbol{In'}^{do})$, thereby detecting path (ii).

Since the BC ensures that all TEEs can access the same set of requests, every TEE could execute any subproblems in each market clearing phase. Then, supported by the countermeasure of the TEAR-DO mechanism, once a subproblem fails to be executed due to Byzantine attacks, each honest TEE will re-execute it. Thus, the issue that a subproblem allocated to κ Byzantine nodes is resolved and the optimal convergence of DO algorithms is achieved.

6.7 Simulation Study

6.7.1 Simulation Setup

Consistent with the models in [232, 237, 241], this paper models buyer utility and seller cost as quadratic functions. The parameterisation of these functions – including their coefficients and the demand/supply ranges – is calibrated following the principles outlined in these references. All tuning parameters used for convergence criteria are set to be 10^{-3}. The time-related performance of the framework will be based on the measurements of communication latency and computation latency. A one round communication latency between two nodes (denoted as L_{cmm}) is defined as the time spent for a receiver to successfully receive a message sent from a sender. L_{cmm} comprises the transmission delay Δ_m, and the propagation delay Δ_p. $\Delta_m = \frac{Ms}{r^{\mathrm{b}}}$ is the time required to place an entire data packet onto the physical transmission medium, where Ms denotes the message size in megabytes (MB) and r^{b} presents the downloading/uploading bandwidth rate. Δ_p is the time it takes for the first bit of that packet to geographically travel from the sender to the receiver. In the rest of this paper, r^{b} and Δ_p are set to be 20MB/s and 30ms, respectively. The computation latency, such as the time required to execute DO algorithms, are measured using Python codes.

In the framework, although Byzantine attacks can be effectively defended,

Byzantine nodes can still formulate an optimal attack strategy to maximise the delay of the iterative DO process. Specifically, consider d subproblems, each of which is assigned to κ Byzantine nodes. If these nodes launch d sequential attacks, they can induce a total delay of $d\Delta$. In this section, the optimal attack strategy is considered in Sections 6.7.4 and 6.7.5 to evaluate the performance of the framework.

We compare the framework against the privacy preserving SSS scheme [241]. The scheme uses the Shamir's secret sharing to ensure privacy-preserving DO. In this scheme, for a single computational operation O with $\boldsymbol{R} = O(\boldsymbol{V})$, its input vector $\boldsymbol{V}$ is encoded into *K* different vectors. Then, *K* executors execute the same operation O with their own unique vectors to output *K* encoded results. Any *k*+1 out of the *K* encoded results can be used to reconstruct the result $\boldsymbol{R}$. Note that the SSS scheme is limited to Byzantine detection rather than Byzantine tolerance, as it is based on a "malicious-with-abort" model [257]. According to this model, the entire DO process must terminate upon the detection of any Byzantine behaviour, thereby precluding optimal convergence. Therefore, the performance comparison between the framework and the SSS scheme is limited to attack-free scenarios. We simulate the framework on a personal computer equipped with an Apple M2 processor.

6.7.2 Evaluation on the Framework under a Meshed Network

We evaluate the framework on the IEEE 30-bus system [258], a partially meshed power network. The evaluation applies the algorithm in [232] to solve the optimisation problem defined by Eq. (6.1) with constraints Eq. (6.2) and Eqs. (6.4a)-(6.4c). The algorithm operates in iterations, each comprising two main steps. First, all buyers and sellers exchange their updated bilateral energy variables. Concurrently, sellers broadcast their updated price-based dual variables to all buyers, and all prosumers send their energy variables to the grid operator. Second, the grid operator sends its updated dual variables to all prosumers. Thus, in the framework, every TEE generates a digital signature and encrypts its computational results in each iteration, as defined in Eq. (6.19)

and Eq. (6.20). Then, each TEE verifies the signatures and decrypts the data from the other B-1 TEEs, as defined in Eqs. (6.21) and (6.22). To implement these operations, the 128-bit AES and the 2048-bit RSA from the PyCryptodome library [259] are adopted. The results in Table 6-1 show that RSA signature generation takes only milliseconds — a substantial reduction compared to the time for finishing a remote attestation [250]. This efficiency makes our signature-based approach highly suitable for the iterative DO-based P2P energy market clearing.

TABLE 6-1. Evaluation of time metrics for computational algorithms

Operation types (12 buyers and 12 sellers)	Time cost (seconds)
128-bit AES Encryption (Eq. (23))	1.3×10^{-5}
128-bit AES Decryption (Eq. (26))	2.2×10^{-5}
2048-bit RSA Signature generation (Eq. (29))	1.1×10^{-3}
2048-bit RSA Signature verification (Eq. (30))	3.1×10^{-4}

TABLE 6-2. Evaluation results of the framework by executing the distributed algorithm in [232] under IEEE 30-bus system

Metrics/situations	12 buyers and 12 sellers	25 buyers and 25 sellers	50 buyers and 50 sellers	100 buyers and 100 sellers
DO algorithm time cost per iteration	6.25×10^{-5} seconds	7.89×10^{-5} seconds	1.19×10^{-4} seconds	4.37×10^{-4} seconds
Communication time per iteration	6.50×10^{-2} seconds	7.11×10^{-2} seconds	9.41×10^{-2} seconds	1.91×10^{-1} seconds
AES time per iteration	4.09×10^{-3} seconds	4.12×10^{-3} seconds	4.23×10^{-3} seconds	5.65×10^{-3} seconds
RSA time per iteration	6.52×10^{-2} seconds	6.52×10^{-2} seconds	6.52×10^{-2} seconds	6.52×10^{-2} seconds
Total iterative rounds	776	1,183	1,629	2,765
Total time (communication time) cost	104.25 (50.44) seconds	166.09 (83.99) seconds	266.42 (153.13) seconds	725.22 (528.12) seconds
Social welfare (centralised solver/DO)	6.908/6.908 ($)	14.594/14.590 ($)	32.581/32.615 ($)	65.940/66.096 ($)

The performance of the framework is then evaluated by executing the DO algorithm [232]. Table 6-2 presents a detailed breakdown of the time costs per iteration, categorised into three components: DO operations, communication, and cryptographic overhead, etc. It is evident that communication is the dominant performance bottleneck compared to the computational time costs. This is because the DO operations are performed directly on plaintexts, and the cryptographic functions (AES/RSA) are computationally lightweight. A deeper analysis of the communication cost reveals its composition and scaling logic. In smaller networks (e.g., 12 buyers, 12 sellers), the overhead is dominated by the propagation delay Δ_p of 60ms. As the network scales to 100 buyers and 100 sellers, the transmission delay Δ_m increases to 131ms, becoming the main contributor. This increase in transmission delay is inherent to the secure design of the framework, which requires an all-to-all broadcast for Byzantine fault tolerance, with a message size that scales with the number of prosumers.

6.7.3 Comparison the Framework with the SSS Scheme

To benchmark the framework, we conduct a performance comparison against the SSS scheme. Both are configured to execute the same DO algorithm [232] for an energy market clearing phase (i.e., $\mathcal{T}_2$ defined in Section 6.3.2). The SSS scheme is implemented using the MP-SPDZ library [257] with a 128-bit security parameter, where *K* represents the number of executors in its multi-party computation setting.

Figure 6-3 illustrates $\mathcal{T}_2$ for both schemes under different prosumer numbers. The results show that the framework is more time-efficient than the SSS scheme across all scenarios. This performance advantage is rooted in its design: the framework relies on highly efficient cryptographic operations (AES for privacy, RSA for Byzantine detection) that consume only milliseconds per iteration as shown in Table 6-2. In contrast, the SSS scheme incurs prohibitive latency for the non-linear operations (e.g., comparison) within the DO algorithm. A single comparison operation, for instance, requires 14-16 rounds of communication among the *K* executors [257]. It is also critical

to note that the SSS evaluation, conducted on a single laptop, does not account for real-world network propagation delays. Including this latency would make the performance gap presented in Figure 6-3 even more pronounced.

Figure 6-3 also reveals that the framework's communication delay scales with the number of prosumers, a finding consistent with the analysis in Table 6-2. While this overhead increases the total execution time, it remains well within acceptable time limits (e.g., under 5 minutes for up to 74 prosumers). Importantly, this communication cost is a necessary trade-off for Byzantine tolerance. As demonstrated in Section 6.5.2, the all-to-all broadcast is the core mechanism that defends against Byzantine attacks and ensures the executed DO algorithm achieves optimal convergence.

6.7.4 Evaluation on the Framework under Byzantine Attacks

While the framework is Byzantine tolerant, the countermeasure of the TEAR-DO mechanism relies on a synchronous model (see Section 6.5.2), which implies that Byzantine attacks can still introduce additional time delays to DO algorithms. To quantify this side effect, we perform a test in which the optimal attack strategy defined in Section 6.7.1 is considered. The simulation is configured to execute the DO algorithm [232] with $M = 50$ buyers and $N = 50$ sellers, creating 101 subproblems per iteration according to the DO algorithm.

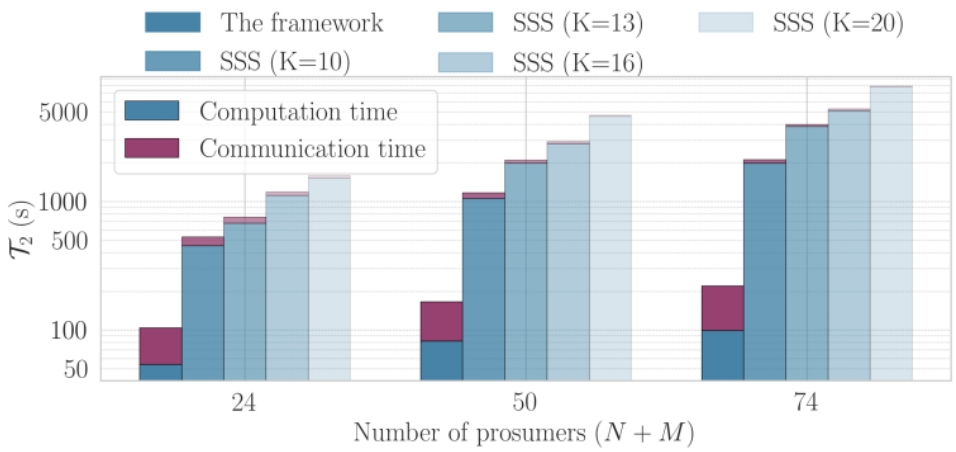

FIGURE 6-3. Evaluation of $\mathcal{T}_2$ with varying number of prosumers.

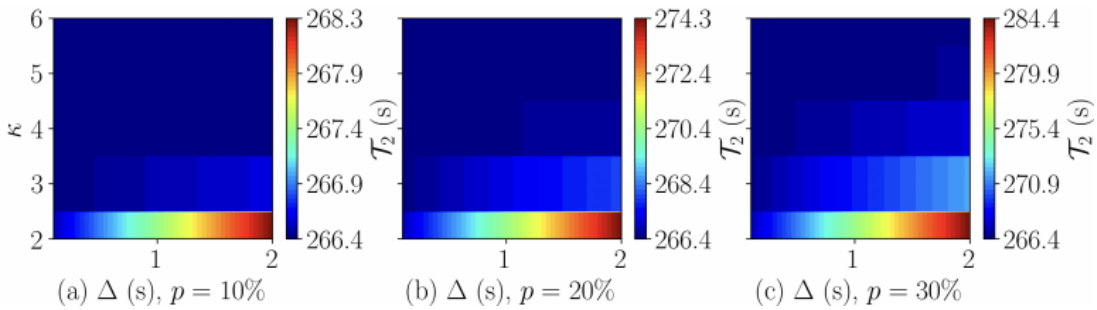

FIGURE 6-4. Evaluation of $\mathcal{T}_2$ across different values of κ, Δ, and p.

Figure 6-4 shows the market clearing time $\mathcal{T}_2$ under this optimal attack strategy, with results plotted against varying synchronous delays Δ, assignment parameters κ, and proportions of Byzantine nodes p. For context, the baseline $\mathcal{T}_2$ without attacks is 266.42 seconds (Table 6-2). As shown in Figure 6-4(c), even when 30% of the nodes (60 nodes) is Byzantine, the additional delay is only 20 seconds for $\Delta = 2$ seconds. This robustness stems directly from the core countermeasure of the TEAR-DO mechanism: an honest TEE that does not receive a valid result for a subproblem within the Δ time-bound will re-execute it. This design guarantees that even a subproblem assigned solely to κ Byzantine nodes can solely incur a maximum delay of Δ. Furthermore, the results in Figure 6-4 also show that by increasing κ, the side effects of Byzantine attacks are further mitigated.

6.7.5 Evaluation on the Redundant Computing in the Framework

While Byzantine attacks are effectively mitigated, the task assignment scheme and the countermeasure of the TEAR-DO mechanism introduce redundant computation. To quantify this overhead, the framework is further evaluated under the optimal attack strategy defined in Section 6.7.1. The simulation is run by executing the DO algorithm [232] with $f = 60$ Byzantine nodes, $M = 50$ buyers, and $N = 50$ sellers.

Figure 6-5(a) shows the distribution of c, the number of times each subproblem is executed by honest nodes on average, for different values of the assignment parameter κ. The data of Figure 6-5(a), averaged over 5,000 trials, reveals that $\kappa = 5$ achieves an optimal computational redundancy — the redundant computation count c is tightly

concentrated around 4. This indicates that for a p=30% Byzantine population ($p = f/B$), a κ of 5 provides sufficient redundancy to handle attacks.

The reasoning for this optimal point involves two competing effects, which can be understood with the help of Figure 6-5(b). On one hand, as κ increases, the probability of a subproblem being assigned exclusively to Byzantine nodes decreases, reducing the number of attack-induced re-executions introduced by the countermeasure. On the other hand, a larger κ also increases the expected number of honest nodes assigned to each subproblem (calculated as $\frac{B-f}{B}\kappa = 0.7\kappa$). The slightly higher average c for $\kappa = 7$ or $\kappa = 10$ compared to $\kappa = 5$ is a result of these two effects. Ultimately, the results in Figure 6-4 and Figure 6-5(a) illustrate that by properly selecting κ, the framework can successfully neutralize Byzantine attacks at the cost of only a well-controlled amount of redundant computations.

6.7.6 Evaluation on the Scalability and the Compatibility of the Framework under a Radial Network

As shown in Table 6-2, first-order DO algorithms like [232] require thousands of iterations to converge, even for a simplified, PTDF-based meshed network model. To demonstrate our framework's ability to reduce the iteration counts, the second-order ALADIN algorithm [254] is adopted. The framework's performance is then evaluated using the ALADIN algorithm on the more complex IEEE 33-bus radial network [260], which involves the multiple physical constraints defined in Eqs. (3a)-(3d).

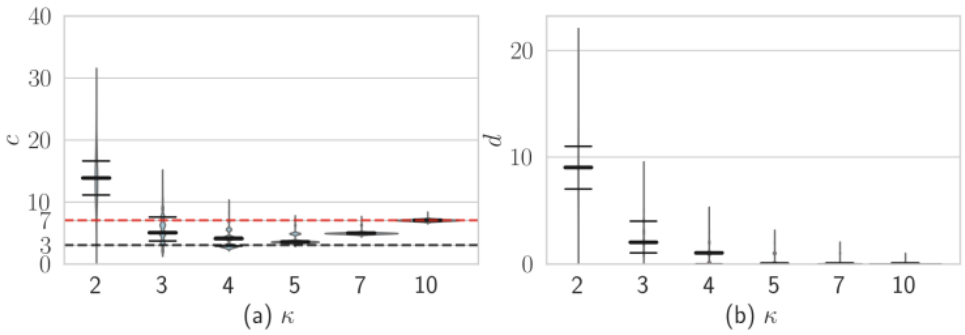

FIGURE 6-5. The influence of κ on c and d.

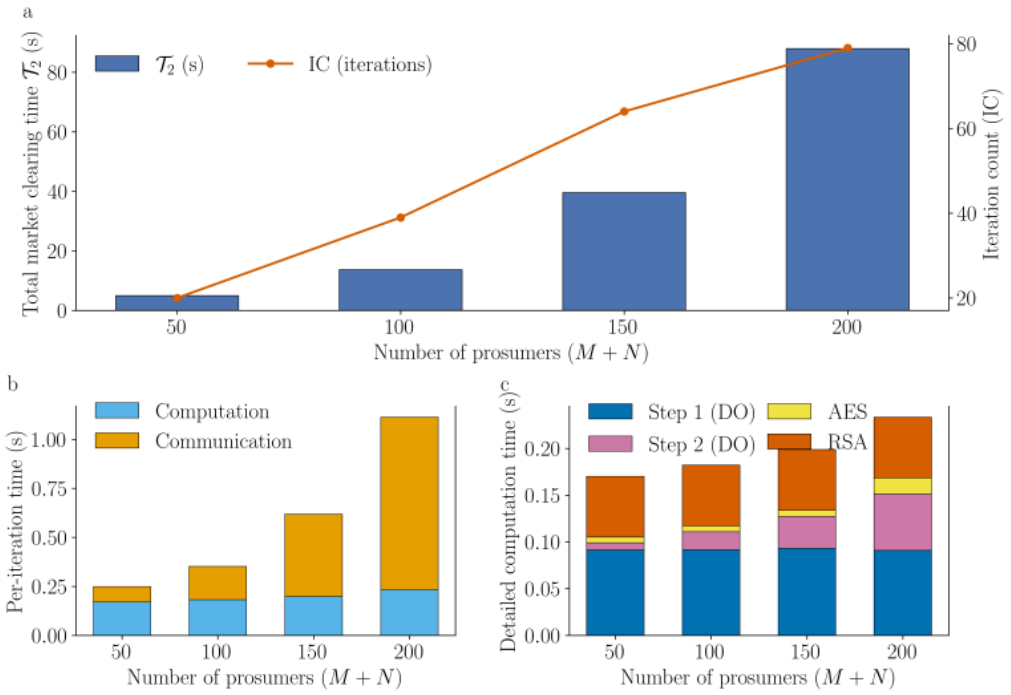

FIGURE 6-6. Evaluation of $\mathcal{T}_2$ across different number of prosumers under IEEE 33-bus system with steps 1 and 2 specified in Section II-B.

Figure 6-6 presents the framework's time-related performance, plotting the total market clearing time $\mathcal{T}_2$ and iteration count (IC) as the number of prosumers scales up to 200, i.e., a total of 10,000 trading pairs when M=N=100. The results in Figure 6-6(a) demonstrate that even with 200 prosumers, $\mathcal{T}_2$ remains below 86 seconds, converging in fewer than 80 iterations. While the dramatic reduction in iteration count is a direct benefit of the ALADIN algorithm's properties, a deeper analysis, presented in Figure 6-6(b), reveals why $\mathcal{T}_2$ remains low despite the increased complexity of ALADIN. Figure 6-6(b), which breaks down the per-iteration cost, confirms that communication remains the dominant bottleneck due to the all-to-all communication designed in the TEAR-DO mechanism, a finding consistent with our earlier evaluations in Table 6-2 and Figure 6-3.

Figure 6-6(c) decomposes the computational component. The ALADIN algorithm is more computationally intensive than the first-order method in [232], but its computation time remains around 100-220 milliseconds per iteration in the tested

setting and is lower than the communication delay. This result supports the use of TEEs for executing DO operations on plaintexts: computation is not the dominant bottleneck in the evaluated cases. The cryptographic overhead introduced by the framework also remains small over the tested prosumer numbers. This is because AES operations require only 10^{-5}-10^{-4} seconds per operation, and the RSA signature workload depends mainly on the number of selected TEEs rather than the number of prosumers. These observations support the framework's compatibility with more complex DO algorithms.

6.8 Conclusion

This chapter addresses the identified trilemma in P2P energy markets: real-time market clearing, privacy preservation for sensitive energy data and optimal convergence under Byzantine attacks. To address this trilemma, the chapter proposes a TEE-based framework with two technical components: the D-TASK protocol for secure and decentralised deployment of a shared secret and an application script, and the TEAR-DO mechanism for time-efficient and attack-resilient DO.

The theoretical analysis shows that, under the stated assumptions, the framework preserves data privacy and tolerates the Byzantine attacks considered in this chapter. The simulation results on IEEE benchmark systems show that the framework improves market-clearing time relative to the compared privacy-preserving scheme while retaining Byzantine resilience. By using lightweight signature-based proofs and executing DO operations on plaintexts inside TEEs, the framework achieves up to an order-of-magnitude speedup in the evaluated settings. The simulations also show that the TEAR-DO countermeasure limits the delay caused by the optimal Byzantine attack strategy with a controlled increase in redundant computation. Its compatibility with the second-order DO algorithm ALADIN indicates that the framework can support faster convergence than first-order DO methods in the tested P2P energy market clearing scenarios.

These results rely on the hardware assumption that each BC node is equipped with

a TEE. In practice, some participating BC nodes may lack TEE support. A conservative fallback is to run D-TASK and TEAR-DO among a TEE-equipped subset of BC nodes, while the remaining BC nodes continue to record blocks and maintain data for each market clearing interval. This fallback preserves the analysis only when the TEE-equipped subset still contains enough honest TEEs to satisfy the Byzantine-tolerance requirements in this chapter. Otherwise, the privacy-preserving and Byzantine-tolerant properties analysed here no longer directly apply. How to guarantee an adequate proportion of honest TEEs within this small set remains a topic for future research.

Another practical limitation concerns the Δ-synchronous condition stated in Section 6.3.1. In the current framework, Δ is a conservative time bound for detecting missing valid TEE results. If the actual network delay temporarily exceeds Δ, a correct TEE may be treated as unresponsive. The TEAR-DO countermeasure then recomputes the missing variables through honest TEEs. This preserves the BC-recorded requests and transactions, but it can increase redundant computation and may cause the market clearing result of that interval to miss the real-time deadline. Such a delay does not by itself invalidate the requests already committed to the BC; rather, it indicates that the interval requires a contingency action, such as postponing settlement or re-running clearing with updated admissible requests under the market rule. Future work can reduce dependence on a fixed Δ by instantiating the BC consensus protocol with asynchronous LBFT protocols, as discussed in Chapter 4. It should also redesign D-TASK and TEAR-DO so that protocol progress relies on eventual message delivery. Such a redesign would target privacy-preserving and Byzantine-tolerant DO under network delays that exceed Δ.

D-TASK is executed offline before repeated market clearing intervals, while TEAR-DO is the main online scalability concern. As the numbers of buyers and sellers increase, encrypted requests, coupling variables and DO subproblems also increase. If the system increases the number of TEEs (i.e., B), although each selected TEE executes fewer subproblems in each iteration, the all-to-all broadcast in Section 6.5.2 then creates larger communication cost among selected TEEs. If B remains small, each selected TEE

executes more subproblems and processes larger outputs and proofs. TEE-side computation can then become the bottleneck. Market expansion therefore creates a computation-communication trade-off in the online TEAR-DO mechanism. Future work should reduce this trade-off while retaining the framework's Byzantine-tolerant property.

CHAPTER 7

Federated Learning System for Power Load Forecasting

Federated learning (FL) for power load forecasting is crippled by an inherent trade-off between Byzantine security and communication efficiency. Existing FL frameworks suffer from a communicational bottleneck at the central server side and are vulnerable to Byzantine attacks. To solve the communication bottleneck, we propose CTP-FL, a new FL paradigm in which "C"ommunication overhead undertaken by a central aggregator is "T"ransferred to all "P"articipants. The transferring is achieved via a multi-blockchain architecture with novel client-to-blockchain, blockchain-to-client, and client-to-client communication patterns. The Byzantine tolerance of the CTP-FL system is further ensured by the two core protocols: a message coding protocol for data recoverability, and a dual consensus protocol for secure and private model aggregation. Simulations confirm that the CTP-FL system reduces communication latency at least 14.48 times compared to current FL paradigms. The CTP-FL paradigm may be of independent interest for a wide range of applications beyond power load forecasting.

7.1 Introduction

The widespread adoption of distributed energy resources (DERs) is fundamentally shifting the power grid from a unidirectional, centralised topology to a multidirectional, decentralised network. This shift introduces a challenge for traditional power load forecasting (PLF), which must now accurately and timely predict load demand amidst the highly variable and intermittent DERs. While machine learning has emerged as an indispensable tool for enhancing PLF, its centralised architecture makes it prone to both privacy breaches and single point of failure [261].

Recently, federated Learning (FL) has gained prominence as a distributed machine

learning paradigm designed to address data privacy challenges [262]. In the FL framework, the machine learning model is trained collaboratively across multiple clients. Each client trains the model locally through its private dataset, producing a local model. These local models are then aggregated by a central server to update a global model, avoiding the need to expose private dataset. Despite this merit, the FL framework still needs to tackle the following two issues for their practical adoption on PLF:

Security issue: One such vulnerability is Byzantine attacks at both client and central server sides [263-266]. On the one hand, malicious clients can exploit the public availability of local model parameters to craft and submit poisoned local models, which strategically disrupts the global model's convergence [263, 264]. On the other hand, the central server could become the target of adversarial attacks [265]. Even more concerning, if the central server itself is Byzantine, it can perform arbitrary malicious actions to the model aggregation process [266].

Time efficiency issue: Numerous studies highlight that, to maintain predictive accuracy, machine learning models for PLF require frequent retraining and update to timely capture the variability introduced by intermittent DERs [267-269]. However, timely updating PLF model remains challenging under the current two FL paradigms, i.e., centralised FL [263] and blockchain (BC)-based FL [270]. In both paradigms, hundreds of clients must transmit megabyte-sized local models to a central aggregator, i.e., a central server or a BC leader node, in each training iteration. This process imposes heavy downloading bandwidth demands on the central aggregator. Moreover, the central aggregator must distribute the updated global model to all clients by the end of each iteration, which makes its uploading bandwidth a significant bottleneck.

This chapter focuses on an operating regime with many clients, repeated federated learning iterations and non-negligible transmitted model or update sizes. It also requires local-model-parameter privacy and Byzantine tolerance at both the client and BC-node sides. In a standard server-based FL protocol, let C denote the number of clients and let S denote the serialised size of one local model or model update in MB. In every

iteration, the central server receives C local models or updates and sends the global model back to C clients. Its server-side traffic is therefore approximately $2CS$ MB per iteration, before considering protocol metadata or retransmission. This traffic is determined by the transmitted model/update size and the number of clients. For example, when C=100 and S=100 MB, the central server handles about 20 GB of ingress and egress traffic in one iteration. When C=200 and S=800 MB, this traffic becomes about 320 GB. Therefore, the motivation for Communication Burden Transfers to all Participants-based Federated Learning (CTP-FL) is the large-scale PLF setting considered in this chapter, where the server-side communication burden and Byzantine-tolerance requirements become non-negligible. For other scenarios in which S is small and C is modest, conventional server-based FL, such as FedAvg [298], may be sufficient, and these scenarios are beyond the chapter's scope.

7.1.1 State-of-the-Art Solutions on the Security and Time Efficiency Issues of the FL Frameworks

Security aspect: One prominent line of research addresses the secure vulnerabilities of the central aggregator by integrating FL with BC technology [271, 272]. In merit of BC's trustworthy feature, [270] uses BC to replace the central aggregator to eliminate the server side's single point of failure. [273] leverages BC's immutability property to record the hash of local and global models, ensuring any incorrect models can be traced back and held accountable afterwards. [274] designs a BC-based incentive mechanism that rewards clients who generate high-quality local models. Despite these merits, BC-based methods introduce huge communication burdens due to BC's consensus process. The process requires a BC leader node to send a block that contains multiple local models to all other BC nodes [270], which incurs high communication delays. More seriously, BC-based schemes are inherently incapable of defending against client-side Byzantine attacks [275].

Another fundamental security vulnerability arises from the public nature of local

model parameters in FL frameworks. By analysing the plaintext local model parameters of honest clients, Byzantine clients can craft well-designed models to evade anomalies detection mechanisms and even controllably bias global models' predictive directions [263]. To tackle this security issue, a reasonable approach is to encrypt local models, thereby preventing model parameters from malicious analysis. However, model aggregation methods for evaluating encrypted local models are computationally intensive, as nonlinear operations (e.g., homomorphic multiplication) essential for encryption-based Byzantine tolerant aggregation methods are notoriously inefficient over ciphertexts [276], which aggravates the time efficiency issue of FL frameworks.

Time efficiency aspect: While reducing local model training time is an active research area, communication latency is widely recognised as the primary bottleneck in large-scale FL systems [271]. This bottleneck is rooted in two fundamental constraints. First, a physical disparity exists where modern computational speeds (Ops/s) are around 10^4 times greater than network bandwidth (bytes/s) [277]. Second, the centralised FL topology dictates that a single aggregator must handle both the massive ingress of local model uploads from all clients and the egress of global model broadcasts to all clients [271]. Although existing approaches mitigate the aggregator's bottleneck by compressing the transmitted data via techniques like quantisation [278], sparsification [279], or model pruning [280], these methods do not circumvent the two constraints. Thus, tackling the aggregator's communication bottleneck by circumventing the two constraints remains a largely unexplored direction, which is a key focus of this paper.

We then ask an unresolved question: *whether there exists a FL paradigm that (i) ensures Byzantine tolerance at both client and server sides while maintaining stringent local model privacy and controllable operational overhead, and (ii) achieves high time efficiency under the two constraints*?

7.1.2 Contributions

This paper answers this question by proposing a new FL paradigm – CTP-FL. The kernel of CTP-FL is to securely dismantle the central aggregator's communication bottleneck caused by the two constraints. It achieves this by introducing a principle "**C**ommunication Burden **T**ransfers to all **P**articipants", which utilises controllable computation overhead to redistribute the communication burdens solely undertaken by the central aggregator to all FL participants.

The CTP-FL paradigm has three distinct features compared to the two FL paradigms. First, in the Client-to-Blockchain (C2B) stage, each client partitions its local model and transmits only a small fragment of its model to each BC node, thereby replacing the central aggregator's ingress bottleneck with a distributed download load across all BC nodes. Second, during the Blockchain-to-Client (B2C) stage, each BC node disseminates just one piece of a global model fragment – a small piece of a global model – to all clients, thus eliminating the central aggregator's broadcast bottleneck. Third, a novel Client-to-Client (C2C) stage is introduced, which further leverages the idle upload-bandwidth of all clients to enhance overall communication efficiency of FL model training.

The CTP-FL's distributed architecture, however, introduces a key design challenge: *how to guarantee data privacy and model recoverability in the FL paradigm where local model fragments are scattered across a network of untrusted BC nodes*? Answering this requires a holistic security strategy that ensures Byzantine tolerance at both client and server sides without imposing prohibitive communication and computation overhead. We achieve this via a synergistic pair of protocols:

1. The message coding protocol. This protocol is designed to secure the fragmented model data throughout its journey. It addresses the following three distinct threats in the C2B stage with a 3-layered defence: (i) Privacy threat: Each local model fragment – a small piece of a local model – is encrypted, preventing parameters' exposure. (ii) Server-side recoverability threat: An erasure coding (EC) algorithm [281] transforms

each local model fragment into multiple data pieces, ensuring each local model fragment can be recovered afterwards even if a fraction of BC nodes is Byzantine. (iii) Client-side recoverability threat: A commitment (PBPC) scheme [282] is applied, allowing BC nodes to verify that each received data piece is a correct part of the original local model fragment, thus thwarting attacks from Byzantine clients who transmit invalid data pieces to prevent fragment recovery. This 3-layered defence is further extended with a proposed double EC strategy to secure the B2C and C2C stages, ensuring all clients can recover a global model in each FL training iteration.

2. The dual consensus protocol. This protocol provides a secure and private engine for aggregating the encrypted local model fragments. It employs a two-stage process: (i) A leaderless protocol: Supported by the EC and PBPC schemes, this protocol acts as a filter to ensure that all BC nodes efficiently reach a consensus on the set of selected clients whose encrypted model fragments are verified as recoverable. (ii) Leader-based protocol: A designated BC leader, operating within a secure and private environment, decrypts the selected, encrypted local model fragments and aggregates them in plaintext to output a global model fragment via a Byzantine-tolerant algorithm [263]. The leader then initiates a consensus on the hash value of the global model fragment. This stage ensures all BC nodes reach an agreement on the hash value.

As a supplementary contribution, comprehensive simulations are conducted to evaluate the performance of the CTP-FL system by comparing it with the two FL paradigms. Three key results are yielded: (i) the communication time cost of the CTP-FL system is at least 14.18 times smaller than that of the two FL paradigms; (ii) the additional computation time cost introduced in the CTP-FL system is well-controlled and no larger than 32 seconds even at the worst situation; (iii) the CTP-FL system is robust to defend against Byzantine attacks launched by both clients and BC nodes. These findings collectively confirm that the CTP-FL system is a practical solution for solving the secure and time-efficient issues existing in training PLF machine learning models.

The remainder of this paper is organised as follows. Section 7.2 introduces the

background of the CTP-FL system. Sections 7.3 and 7.4 present the technical details of the proposed two protocols. Section 7.5 reports and discusses the simulation results. Section 7.6 concludes the paper.

7.2 Background

This section first introduces the five technologies. Then, an overview of the CTP-FL system is presented, followed by describing its security model and goals.

7.2.1 Erasure Coding Algorithm

EC is a data encoding/decoding technique that divides data into multiple small fragments and enables data recovery from a subset of these fragments [281]. This technology is used to achieve the Byzantine tolerance ability at the server (BC) side.

Setup: Let $\mathbb{F}_q$ be a finite field of large size q, where q is a prime. For a positive integer T, a piece of data $w \in \mathbb{F}_q$ is divided into T field values $w = w_0|w_1|\dots|w_{T-1}$, where "|" is the concatenation operation. These T field values are then interpreted as the coefficients of a polynomial $P_w(X)$ over $\mathbb{F}_q$:

$$P_w(X) = \sum_{t=0}^{T-1} w_t X^t \tag{7.1}$$

Encoding process: The encoding procedure transforms the polynomial $P_w(X)$ into K polynomial outputs via Eq. (7.2). Specifically, K distinct evaluation points $\{\alpha_1, \alpha_2, \dots, \alpha_K\}$ in $\mathbb{F}_q$ are chosen ($K > T$) to generate K polynomial outputs via Eq. (7.1):

$$\{P_w(\alpha_1), P_w(\alpha_2), \dots, P_w(\alpha_k), \dots, P_w(\alpha_K)\} \tag{7.2}$$

Decoding process: Given any subset of T correct values $\{P_w(\alpha_{k_t})|k_t \in [K]\}$ among the *K* values in Eq. (7.2), $P_w(X)$ can be recovered via Eq. (7.3), where $[K]$ is short for the set $\{1,2,\dots,K\}$:

$$P_w(X) = \sum_{k_t \in [K], t \in [T]} P_w(\alpha_{k_t}) \cdot \mathcal{L}_{k_t}(X) \tag{7.3}$$

where $\mathcal{L}_{k_t}(X)$ is the Lagrange basis polynomial defined by Eq. (7.4):

$$\mathcal{L}_{k_t}(X) = \prod_{j_t \in [K], j_t \neq k_t} \frac{X - \alpha_{j_t}}{\alpha_{k_t} - \alpha_{j_t}} \tag{7.4}$$

Then, directly extracting the T polynomial coefficients of $P_w(X)$ yields the original data $w = w_0|w_1|\ldots|w_{T-1}$. Notably, Byzantine clients can maliciously craft their fragments without following the operations of Eq. (7.1) and Eq. (7.2), making the recovery operations, i.e., Eq. (7.3) and Eq. (7.4), unable to work. To detect this misbehaviour, the commitment scheme is introduced.

7.2.2 Position-Binding Polynomial Commitment

The position-binding polynomial commitment (PBPC) scheme [282] enables a committer to generate an unforgeable commitment for a polynomial, i.e., $P_w(X)$ in this work. Using this commitment, the scheme allows the committer to attest the correctness of each field value in a vector via generating a proof. The details of the PBPC scheme are shown as below:

Setup: Let λ denote the security parameter. Let G_1 and G_2 be a cyclic group of prime order q and let G_T be a target group such that there is an non-degenerate bilinear pairing [283]: $e: G_1 \times G_2 \rightarrow G_T$, where g_1 and g_2 are the generators of G_1 and G_2, respectively. A secret $\alpha \in \mathbb{F}_q$ is selected, which is not revealed to any participant. Using α, the Structured Reference String (*SRS*) is generated as follow:

$$SRS = \left\{g_1, {g_1}^{\alpha}, {g_1}^{\alpha^2}, \ldots, {g_1}^{\alpha^K}; g_2, {g_2}^{\alpha}, {g_2}^{\alpha^2}, \ldots, {g_2}^{\alpha^K}\right\} \tag{7.5}$$

$pp = (\lambda, q, G_1, G_2, g_1, g_2, e(\cdot,\cdot), SRS)$ are public parameters. The PBPC scheme includes the following three components:

Commitment generation: For a vector with its values being $\{P_w(\alpha_1), P_w(\alpha_2), \ldots, P_w(\alpha_k), \ldots, P_w(\alpha_K)\}$ defined in Eq. (7.2), a commitment VC committed for $P_w(X)$ at point α is computed through a Multi-Scalar Multiplication (MSM) operation [284]:

$$VC = {g_1}^{P_w(\alpha)} \tag{7.6}$$

Proof generation for the k^{th} *value* $P_w(\alpha_k)$ *of the vector*: The committer produces a polynomial $P_k(X)$ for $P_k(X) = \frac{P_w(X)-P_w(\alpha_k)}{X-\alpha_k}$ and constructs the proof π_k for the k^{th} value:

$$\pi_k = {g_1}^{P_k(\alpha)}, k \in [K] \tag{7.7}$$

Verification of the k^{th} *value in the vector*: taking $pp = (\lambda, q, G_1, G_2, g_1, g_2, e(\cdot,\cdot), SRS)$, the index k, the commitment VC, the k^{th} field value $P_w(\alpha_k)$ and the proof π_k as inputs, a verifier checks whether the equation of Eq. (7.8) holds:

$$e\left(VC/{g_1}^{P_w(\alpha_k)}, g_2\right) = e(\pi_k, {g_2}^{\alpha-\alpha_k}) \tag{7.8}$$

If this holds, π_k attests that $P_w(\alpha_k)$ is the k^{th} value of the vector in Eq. (7.2) corresponding to the polynomial $P_w(X)$ in Eq. (7.1). The PBPC scheme is used to detect Byzantine client devices who generate vectors without following the operation in Eq. (7.2).

7.2.3 Other Technologies

The message encryption and decryption (MED), the Merkle proofs, the Trusted Execution Environments (TEEs) and the remote attestations are also introduced. The MED scheme and the TEEs ensure the privacy of the local models' parameters. The Merkle proofs and the remote attestations are utilised in the proposed dual consensus protocol to safeguard its security.

MED scheme. Let a secret-public key pair be (*SK*, *PK*). For an arbitrary message m, the encryption and decryption operations are defined in Eq. (7.9) and Eq. (7.10), respectively:

$$m^{\text{E}} = Enc(m, PK) \tag{7.9}$$

$$m = Dec(m^{\text{E}}, SK) \tag{7.10}$$

where $Enc(\cdot)$ and $Dec(\cdot)$ are the encryption and decryption algorithms, respectively, m^{E} is the encryption version of m.

Merkle proof scheme. First, given *N* pieces of data $\{x_1, x_2, \ldots, x_N\}$, the Merkle root generation algorithm $Merkr(\cdot)$ is applied to output a root hash h^{r}:

$$h^{\mathrm{r}} = Merkr(x_1, x_2, \dots, x_N) \tag{7.11}$$

Then, the Merkle proof generation algorithm $MP(\cdot)$ is applied to generate a Merkle proof mp_n for x_n:

$$mp_n = MP(x_n, h^{\mathrm{r}}), n \in [1,2, \dots, N] \tag{7.12}$$

Finally, the Merkle proof verification algorithm $MPV(\cdot)$ is used to verify the validity of a Merkle proof mp_n:

$$\{0,1\} = MPV(x_n, h^{\mathrm{r}}, mp_n), n \in [1,2, \dots, N] \tag{7.13}$$

if $MPV(\cdot)$ outputs 1, mp_n is valid, otherwise, mp_n is invalid.

TEEs and remote attestation RA. This paper uses TEEs to process sensitive local model parameters, while using RA to attest the correctness of the executed operations in TEEs. A TEE is a processer area, the data in which is inaccessible by outside operating system [285]. Let an attestation generation-verification key pair be (*GK*, *VK*). For an evidence E generated by a TEE, the attestation generation (verification) algorithm $RAG(\cdot)$ ($RAV(\cdot)$) is defined as follows:

$$RA = RAG(E, GK) \tag{7.14}$$

$$\{0,1\} = RAV(RA, VK) \tag{7.15}$$

A well-known limitation of TEEs is their limited memory [286, 287]. Under this constraint, it is impractical to execute memory-intensive model aggregation operations within TEEs. To tackle this limitation, a multi-BC architecture is integrated into the CTP-FL system, making each TEE only need to aggregate local model fragments. The overview of the multi-BC-based CTP-FL system is presented as follows.

7.2.4 System Architecture

In the CTP-FL system, B BCs and C clients are considered. The b^{th} BC is denoted as $\{BC_b\}_{b\in[B]}$. Each BC has K BC nodes, and the ${k_b}^{th}$ node within BC_b is denoted as $\{N_{k_b}\}_{k_b\in[K_b]}$ with $K_b = K$ for every $b \in [B]$. Each BC node configures a TEE in which a shared secret key *SK* is stored, which can be achieved through the protocol in [285]. The FL training process progresses in iterative rounds, and each round includes

4 phases, as shown in Figure 7-1. At the e^{th} iteration ($e \geq 1$), each client device trains on a global model $\boldsymbol{\theta}_{e-1}^{\mathrm{g}}$, where g is the tag of global models. When $e = 1$, $\boldsymbol{\theta}_{0}^{\mathrm{g}}$ is an initial global model. The summary of each phase is presented as follows.

Phase 1. At the start of the e^{th} iteration t_e, the client device D_c ($\forall c \in [C]$) trains on $\boldsymbol{\theta}_{e-1}^{\mathrm{g}}$ using its private dataset, yielding a local model $\boldsymbol{\theta}_{e,c}^{\mathrm{l}}$, where l denotes the tag of local models. D_c uses the proposed message coding protocol (see Section 7.3) to handle $\boldsymbol{\theta}_{e,c}^{\mathrm{l}}$, generating $B \times K$ local model subfragments, where a local model subfragment is a small piece of a full local model. The total time cost of this phase is Δ_e^{Loc1}.

Phase 2. Each client device disseminates its $B \times K$ local model subfragments to the $B \times K$ BC nodes following the instructions defined in the proposed message coding protocol. The total time cost of this phase is Δ_e^{C2B}.

Phase 3. Each of the B BCs generates a global model fragment via the proposed dual consensus protocol (see Section 7.4). The time cost of this phase is Δ_e^{C}.

Phase 4. The proposed double EC strategy is applied to enable every client to reconstruct the global model $\boldsymbol{\theta}_{e}^{\mathrm{g}}$. The total time cost of this phase is $\Delta_e^{\mathrm{B2C}} + \Delta_e^{\mathrm{C2C}} + \Delta_e^{\mathrm{Loc2}}$.

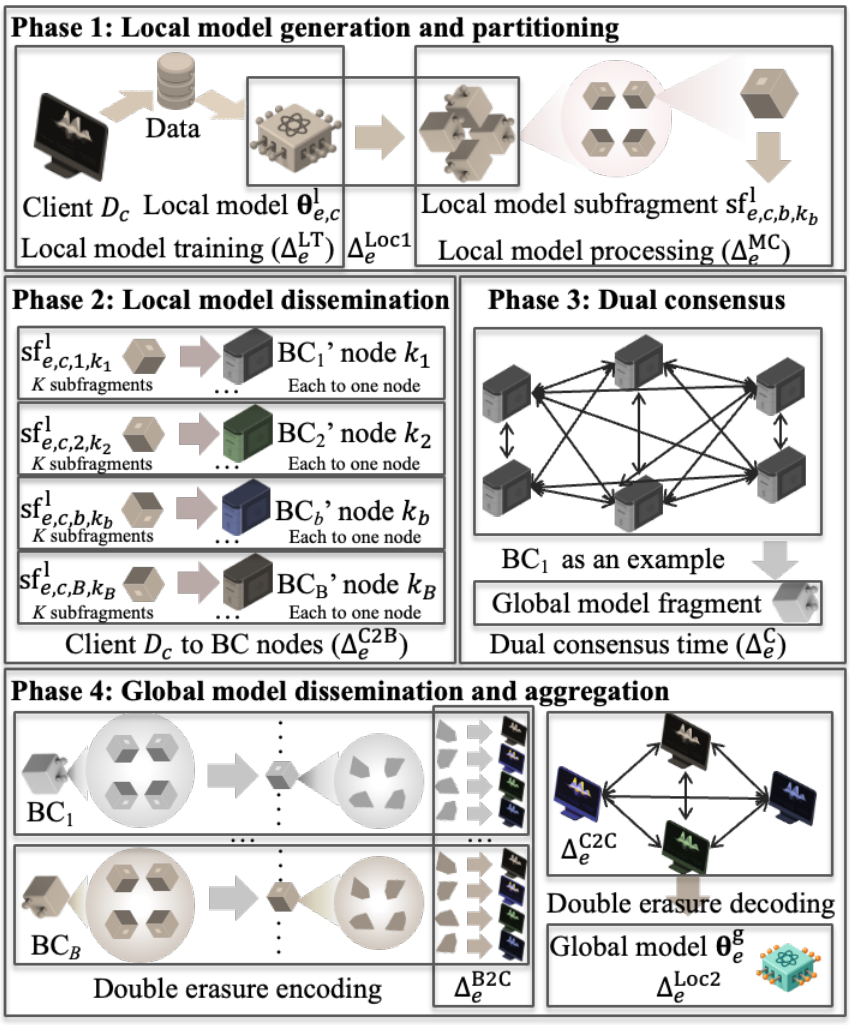

FIGURE 7-1. The four-phase FL model training process in an iteration.

The multi-BC design in CTP-FL does not require extra cross-chain maintenance or cross-chain cryptographic material generation during an FL iteration. Instead, this design reduces the communication and aggregation load assigned to each BC. Specifically, during an FL training iteration, the B BCs do not exchange blocks, transactions, proofs or cryptographic material. Each BC is assigned one model-fragment index b and processes only local model fragments with that index. The message coding protocol further partitions each local model $\boldsymbol{\theta}^{l}_{e,c}$ into B fragments, and only the b-th local model fragment is sent to the blockchain BC_b. With this design, BC_b filters and only aggregates the b-th local model fragments and outputs the b-th global model fragment. Each BC therefore completes fragment aggregation

independently, rather than maintaining inter-BC state or cross-chain messages. This design reduces the per-BC message size from *S* to approximately *S/B* and avoids requiring one TEE to aggregate a full local model. These two effects address the communication bottleneck in Section 7.1.1 and the TEE memory limit in Section 7.2.4.

7.2.5 System Model and Goals

In the proposed system, the Byzantine corruption model and Δ-synchronous network model are considered:

Byzantine corruption model: For the $K = T + f^{\mathrm{By}}$ nodes in a BC, there are up to f^{By} Byzantine BC nodes with $T = 2f^{\mathrm{By}} + 1$. For the $C = 3f^{\mathrm{Cli}} + 1$ clients, there are up to f^{Cli} Byzantine clients. Byzantine participants can perform any malicious actions to disrupt the FL training process.

Δ-*synchronous network model:* In the FL network, any message sent at time *t* is guaranteed to be received by the receiver by time $t + \Delta$, where the "Δ" denotes a time duration.

The CTP-FL system is designed to fulfill the four key objectives implied in the asked question (see Section 7.1.1):

1) The privacy of local model parameters is preserved.

2) The communication time cost in each FL model training iteration is reduced compared to that of the two FL-paradigms, i.e., centralised FL and BC-FL paradigms.

3) The CTP-FL system can defend against attacks launched by Byzantine client devices and/or Byzantine BC nodes.

4) The introduced computation time cost is largely smaller than the local model training time.

The proposed system addresses the four goals by integrating the message coding protocol and the dual consensus protocol. Sections 7.3 and 7.4 detail these protocols. The aggregation function executed in the leader's TEE can be selected according to the threat model. For a setting that only requires weighted averaging and does not evaluate

model poisoning attacks, the leader's TEE can use FedAvg [298]. In the simulations, the aggregation operation is the Byzantine-tolerant trimmed mean aggregation (TMA) algorithm [263], because Section 7.5.7 evaluates untargeted model poisoning attacks. Thus, FedAvg and related methods for small models are out of this chapter's scope.

7.3 Message Coding Protocol

This section presents the message coding protocol Π^{MC} and its core objective is to securely (i) transform each client's large local model into a set of small, encrypted, verifiable subfragments, and (ii) transform the global model subfragments generated by each BC back to the global model fragments. By introducing additional computational tasks, this transformation design is the key to evenly redistributing communication load across the network of all FL participants while upholding stringent security guarantees. As shown in Figure 7-2, Π^{MC} governs the entire data lifecycle of the three stages, i.e., the client-to-blockchain node (C2B) stage, the blockchain node-to-client (B2C) stage and the client-to-client (C2C) stage. The detailed computational operations designed for the three stages are presented below.

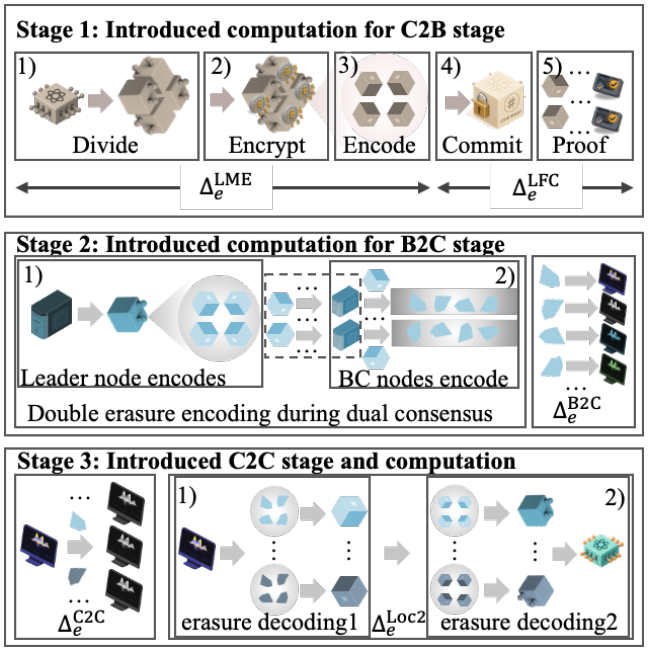

FIGURE 7-2. The 3-stage communication workflows.

7.3.1 Computations in the Client-to-Blockchain Node Stage

The details of BC and TEEs are presented as follows. The two layered procedures are designed to securely reduce the communication time cost (Δ_e^{C2B}) of the C2B stage:

Local model encoding procedure. This procedure divides a local model into B encrypted fragments to preserve the model's privacy. It then uses EC encoding algorithm to divide each encrypted fragment, ensuring the recoverability of each encrypted fragment when considering Byzantine BC nodes.

1) The client device D_c ($\forall c \in [C]$) evenly partitions $\mathbf{\theta}_{e,c}^{\mathrm{l}}$ into B local model fragments $\{\mathbf{f}_{e,c,1}^{\mathrm{l}},\ldots, \mathbf{f}_{e,c,b}^{\mathrm{l}},\ldots, \mathbf{f}_{e,c,B}^{\mathrm{l}}\}$, where $\mathbf{f}_{e,c,b}^{\mathrm{l}}$ is the b^{th} local model fragment with respect to $\mathbf{\theta}_{e,c}^{\mathrm{l}}$.

2) D_c encrypts the fragment $\mathbf{f}_{e,c,b}^{\mathrm{l}}$ ($\forall b \in [B]$) via Eq. (7.9).

3) D_c divides the encrypted $\mathbf{f}_{e,c,b}^{\mathrm{l}}$ ($\forall b \in [B]$) into K_b local model subfragments $\{\mathrm{sf}_{e,c,b,1}^{\mathrm{l}},\ldots, \mathrm{sf}_{e,c,b,k_b}^{\mathrm{l}},\ldots, \mathrm{sf}_{e,c,b,K_b}^{\mathrm{l}}\}$ via Eq. (7.2), where $\mathrm{sf}_{e,c,b,k_b}^{\mathrm{l}}$ is the ${k_b}^{th}$ local model subfragment with respect to the local model fragment $\mathbf{f}_{e,c,b}^{\mathrm{l}}$.

Local fragment commitment procedure. Since Byzantine client devices can arbitrarily crate their subfragments, making the corresponding fragment unable to recover via Eq. (7.3). To tackle this issue, D_c must execute the following two operations:

4) For the encrypted fragment $\mathbf{f}_{e,c,b}^{\mathrm{l}}$ and its K_b local model subfragments $\{\mathrm{sf}_{e,c,b,1}^{\mathrm{l}},\ldots, \mathrm{sf}_{e,c,b,k_b}^{\mathrm{l}},\ldots, \mathrm{sf}_{e,c,b,K_b}^{\mathrm{l}}\}$, the client device D_c generates a commitment $VC_{e,c,b}^{\mathrm{l}}$ for $\mathbf{f}_{e,c,b}^{\mathrm{l}}$ via Eq. (7.6).

5) D_c then executes Eq. (7.7) to generate a proof $\pi_{e,c,b,k_b}^{\mathrm{l}}$ for the local model subfragment $\mathrm{sf}_{e,c,b,k_b}^{\mathrm{l}}$ ($k_b \in [K_b]$).

After executing the above computation operations in 1)-5), D_c constructs

$\sum_{b=1}^{B} K_b = K \times B$ local messages $\{e,\ c,\ VC_{e,c,b}^{\mathrm{l}},\ \mathrm{sf}_{e,c,b,k_b}^{\mathrm{l}},\ \pi_{e,c,b,k_b}^{\mathrm{l}}\}$ for every $b \in [B]$ and $k_b \in [K_b]$. D_c then initiates the C2B stage and sends the local message $\{e,\ c,\ VC_{e,c,b}^{\mathrm{l}},\ \mathrm{sf}_{e,c,b,k_b}^{\mathrm{l}},\ \pi_{e,c,b,k_b}^{\mathrm{l}}\}$ to the BC node N_{k_b}. This local message sending design thus redistributes the downloading bandwidth burdens of a BC leader node to every BC node.

Security analysis. The privacy of local model parameters is guaranteed by executing Eq. (7.9). Two types of attacks can be launched by Byzantine clients: attack 1 – a Byzantine client may generate K_b incorrect local model subfragments, making their encrypted local model fragments irrecoverable via Eq. (7.3), and attack 2 – a Byzantine client may launch untargeted model poisoning attack [263]. The above two procedures solely focus on providing foundations for resolving attack 1.

7.3.2 Computations in the Blockchain Node-to-Client Stage

Let a global model fragment be $\mathbf{f}_{e,b}^{\mathrm{g}}$. The proposed double EC encoding strategy is applied to reduce the communication time cost Δ_e^{B2C} of the B2C stage for $\mathbf{f}_{e,b}^{\mathrm{g}}$ dissemination:

1) In the first EC encoding process, a leader node L_b ($b \in [B]$) in BC_b encodes $\mathbf{f}_{e,b}^{\mathrm{g}}$ into $K = K_b$ global model subfragments $\{\mathrm{sf}_{e,b,1}^{\mathrm{g}}, \ldots, \mathrm{sf}_{e,b,k_b}^{\mathrm{g}}, \ldots, \mathrm{sf}_{e,b,K_b}^{\mathrm{g}}\}$ via Eq. (7.2) and sends $\mathrm{sf}_{e,b,k_b}^{\mathrm{g}}$ to the BC node N_{k_b} ($\forall k_b \in [K_b]$).

2) In the second EC encoding process, the BC node N_{k_b} ($\forall k_b \in [K_b]$) further encodes the received $\mathrm{sf}_{e,b,k_b}^{\mathrm{g}}$ into C min-fragments $\{\mathrm{mf}_{e,b,k_b,1}^{\mathrm{g}}, \ldots, \mathrm{mf}_{e,b,k_b,c}^{\mathrm{g}}, \ldots, \mathrm{mf}_{e,b,k_b,C}^{\mathrm{g}}\}$ via Eq. (7.2).

After executing the operations in 1) and 2), the BC node N_{k_b} ($\forall k_b \in [K_b]$) collectively initiates the B2C stage and sends the mf-message, denoted as $\{e, \mathrm{mf}_{e,b,k_b,c}^{\mathrm{g}}\}$, to the client device D_c. Under this design, the global model fragment

dissemination task originally undertaken by the BC leader node is now evenly allocated to every BC node.

Security analysis. The double EC encoding strategy is the kernel to achieve Byzantine tolerance at the server (BC) side due to the fact – if an encoder is honest, it will faithfully execute Eq. (7.2). Supported by the fact, if a BC leader node L_b is honest, the global model fragment $\mathbf{f}_{e,b}^{\mathrm{g}}$ is recoverable via Eq. (7.3). Since each BC has $2f^{\mathrm{By}}+1$ honest BC nodes, at least $2f^{\mathrm{By}}+1$ global model subfragments are also recoverable via Eq. (7.3). This EC encoding-based design plays a foundation for clients to recover every global model fragment in the C2C stage. The Byzantine leader node issue will be discussed in Section 7.4.

7.3.3 Computations in the Client-to-Client Stage

The C2C stage is uniquely introduced in the C2C-FL system. The C2C stage further transfers a part of uploading bandwidth overheads (for disseminating global model subfragments) undertaken by BC nodes to all clients, which thus introduces the idle upload-bandwidth capacity of clients into the FL training process. Under this design, the C2C stage further balances the uploading bandwidth usage among all participants during the global model dissemination process, thereby improving the overall communication efficiency of FL.

Specifically, the double EC decoding strategy associated to the above-described double EC encoding strategy (see Section 7.3.2) is designed to enable every honest client to recover all global model fragments. The detail of this stage is as follows:

Once the client device D_c $(\forall c \in [C])$ receives the mf-message $\{e,\ \mathrm{mf}_{e,b,k_b,c}^{\mathrm{g}}\}$ $(\forall k_b \in [K_b]$ and $(\forall b \in [B])$ from the BC node N_{k_b}, it stores the mf-message and initiates the C2C stage by sending $\{e,\ \mathrm{mf}_{e,b,k_b,c}^{\mathrm{g}}\}$ to all other clients, which spends a time cost of Δ_e^{C2C}. After that, D_c spends a time cost of Δ_e^{Loc2} to execute the double EC decoding operations:

1) Once the client device D_c receives $2f^{\mathrm{By}}+1$ mf-messages (e.g., $\{e$, $\mathrm{mf}^{\mathrm{g}}_{e,b,k_b,c}\}$) with respect to the global model subfragment $\mathrm{sf}^{\mathrm{g}}_{e,b,k_b}$, it recovers $\mathrm{sf}^{\mathrm{g}}_{e,b,k_b}$ via Eq. (7.3).

2) Upon recovering $2f^{\mathrm{By}}+1$ global model subfragments with respect to the global model fragment $\mathbf{f}^{\mathrm{g}}_{e,b}$, the client device D_c decodes $\mathbf{f}^{\mathrm{g}}_{e,b}$ via Eq. (7.3).

7.4 Dual Consensus Protocol

This section presents the dual consensus protocol designed to solve the challenge within the CTP-FL paradigm: *how to generate a correct global model fragment in each BC when each BC node only holds encrypted local model subfragments due to the design of the C2B stage*. Our solution is a two-stage "filter-then-aggregate" method, which first identifies the recoverable local model fragments and then securely aggregates them. This method is realised via two synergistic sub-protocols. First, a leaderless protocol acts as a filter, enabling all BC nodes in the same BC to collectively identify the set of clients whose encrypted local model fragments are verifiably recoverable. Second, a leader-based protocol performs the aggregation, where a leader, operating within a TEE, securely aggregates the identified fragments.

7.4.1 Leaderless Consensus Protocol

The leaderless consensus protocol Π^{LL} acts as the filter in our "filter-then-aggregate" strategy. It achieves the "filter" goal in a fully decentralised manner by executing the operations defined in the following two sequential stages:

Local message processing stage. This stage determines the validated clients' local model subfragments. To this end, upon receiving local model subfragments from clients at time $t_e+\Delta_e^{\mathrm{Loc1}}+\Delta_e^{\mathrm{C2B}}$ (see Figure 7-1), the BC node N_{k_b} ($\forall k_b \in [K_b]$) spends a duration of Δ_e^{C1} to execute the operations in 1)-3):

1) N_{k_b} inputs the received local messages (e.g., $\{e$, c, $VC^{\mathrm{l}}_{e,c,b}$, $\mathrm{sf}^{\mathrm{l}}_{e,c,b,k_b}$,

$\pi^{\mathrm{l}}_{e,c,b,k_b}\}$ articulated in Section 7.3.1) to its TEE TEE_{k_b}. TEE_{k_b} verifies the validity of each local message via Eq. (7.8). Client IDs (e.g., c) from all successfully validated messages are recorded into a pre-vector, Vec^{P}_{e,k_b}.

2) TEE_{k_b} generates a remote attestation RA^{P}_{e,k_b} for the pre-vector Vec^{P}_{e,k_b} via Eq. (7.14). N_{k_b} broadcasts the pre-vector message $\{Vec^{\mathrm{P}}_{e,k_b}, RA^{\mathrm{P}}_{e,k_b}\}$ to all BC nodes in the b^{th} BC, i.e., BC_b.

3) If N_{k_b} is not the leader node in BC_b, it also sends every received local message, whose client device's ID is included in Vec^{P}_{e,k_b}, to the BC leader node L_b in BC_b.

Security analysis: By verifying PBPC proofs via Eq. (7.8) in 1), each BC node filters out invalid subfragments. The remote attestation RA^{P}_{e,k_b} generated in 2) ensures that if a BC node provides an invalid pre-vector message to other BC nodes, honest BC nodes can detect it afterwards by executing Eq. (7.15). The operation in 3) ensures that the leader BC node can receive enough local model subfragments to recover corresponding local model fragments afterwards via Eq. (7.3).

Vector agreement stage. This stage transfers the validated pre-vectors to a definitive deter-vector of clients whose local model fragments are verifiably recoverable. At time $t_e + \Delta^{\mathrm{Loc}}_e + \Delta^{\mathrm{C2B}}_e + \Delta^{\mathrm{C1}}_e$, each BC node executes the following two-step procedure (i.e., 4) and 5)) with a duration of Δ^{C2}_e:

4) The BC node N_{k_b} ($\forall k_b \in [K_b]$) inputs all received pre-vector messages into its TEE. The TEE selects the pre-vectors whose remote attestations are validated by Eq. (7.15). The TEE then performs a vote tally: for each client ID, it counts its occurrences across all validated pre-vectors. An ID is recorded into a deter-vector Vec^{D}_{e,k_b}, only if its count is no smaller than the Byzantine fault tolerance threshold of $T = 2f^{\mathrm{By}} + 1$. Finally, the TEE generates the remote attestation RA^{D}_{e,k_b} for its generated deter-vector Vec^{D}_{e,k_b}.

5) To improve the communication efficiency of the leaderless consensus protocol, a committee of $p = f^{\mathrm{By}} + 1$ BC nodes is randomly selected using a public random value $RV_{e,b}^{\mathrm{C1}}$ and the previous block hash $BH_{e-1,b}$, as defined in Eq. (7.16) and Eq. (7.17). Only these p selected BC nodes broadcast their generated deter-vector messages to all other BC nodes.

$$\boldsymbol{S}_b = Assign\left(RV_{e,b}^{\mathrm{C1}}, BH_{e-1,b}, p\right) = \underset{|\boldsymbol{S}_b|=p}{\mathrm{Arg\,min}}\left\{h_1, \dots, h_{k_B}\right\} \tag{7.16}$$

$$h_{k_b} = H\left(RV_{e,b}^{\mathrm{C1}} \middle| BH_{e-1,b} \middle| k_b\right), \forall b \in [B], \forall k_b \in [K_b] \tag{7.17}$$

where h_{k_b} is the hash value computed for the BC node N_{k_b}. The algorithm $Assign(\cdot)$ selects the p BC nodes whose hash values rank the first p minimum values in $\{h_1, \dots, h_{k_B}\}$.

Security analysis. In 4), by requiring $2f^{\mathrm{By}} + 1$ attestations for any given client, it ensures that a deter-vector only includes clients for whom enough local model subfragments exist to guarantee full fragment recovery via Eq. (7.3), even in the presence of f^{By} Byzantine nodes per BC. The operations in 5) ensure at least one honest BC node's deter-vector that contains all honest clients' IDs can be timely broadcast and considered.

7.4.2 Leader-based Protocol

After completing the execution of Π^{LL}, the leader-based protocol (Π^{L}) is initiated in each BC to output a correctly aggregated global model fragment. It consists of two stages:

Global model subfragments generation stage. Based on the received deter-vectors (up to $f^{\mathrm{By}} + 1$ vectors), each leader node in corresponding BC uses its TEE to privately generate a global model fragment and corresponding K global model subfragments. Specifically, at the time point $t_e + \Delta_e^{\mathrm{Loc}} + \Delta_e^{\mathrm{C2B}} + \Delta_e^{\mathrm{C1}} + \Delta_e^{\mathrm{C2}}$, the leader node L_b orchestrates a secure process within its TEE over a duration of Δ_e^{C3}. This process follows a "Input-Processing-Output" logic:

Input part. The leader node L_b inputs all received deter-vector messages to its TEE TEE_{L_b} ($b \in [B]$). TEE_{L_b} then verifies the validity of the remote attestations recorded in these received deter-vector messages via Eq. (7.15). For validated deter-vector messages, TEE_{L_b} inputs all IDs appearing in these deter-vectors into an agreement vector Vec^{A}_{e,L_b}.

Processing part. The leader node L_b feeds all stored local model subfragments associated to the IDs in Vec^{A}_{e,L_b} back to TEE_{L_b}. The three operations are sequentially execution: TEE_{L_b} first reconstructs the encrypted local model fragments via Eq. (7.3) and then decrypts these encrypted local model fragments via Eq. (7.10). TEE_{L_b} finally uses the model aggregation algorithm [263] to output a global model fragment $\mathbf{f}^{\mathrm{g}}_{e,b}$.

Output part. TEE_{L_b} executes the first EC encoding operation of the double EC encoding strategy (see Section 7.3.2) to transfer $\mathbf{f}^{\mathrm{g}}_{e,b}$ into K global model subfragments $\{\, \mathrm{sf}^{\mathrm{g}}_{e,b,1}, \ldots, \mathrm{sf}^{\mathrm{g}}_{e,b,k_b}, \ldots, \mathrm{sf}^{\mathrm{g}}_{e,b,K_b} \,\}$. For security purposes, TEE_{L_b} generates the remote attestation RA^{A}_{e,L_b} via Eq. (7.14), the root hash $\mathrm{rh}^{\mathrm{g}}_{e,b}$ via Eq. (7.11) (with the inputs of $Merkr(\cdot)$ being the K subfragments), and the Merkle proof $\mathrm{mp}^{\mathrm{g}}_{e,b,k_b}$ for $\mathrm{sf}^{\mathrm{g}}_{e,b,k_b}$ via Eq. (7.12).

Security analysis. The input part identifies trusted inputs – by verifying remote attestations on the deter-vectors, TEE_{L_b} will only select the IDs of the clients whose local model fragments are verifiably recoverable. The processing part guarantees confidential and trustworthy computation on aggregating a global model fragment, as the entire decoding, decryption, and aggregation process is shielded within the TEE. The use of a Byzantine-tolerant algorithm [263] is to defend against attack 2 specified in Section 7.3.1. The output part produces verifiable outputs used for the next stage, where RA^{A}_{e,L_b} is used to validate the root hash $\mathrm{rh}^{\mathrm{g}}_{e,b}$. The Merkle proof $\mathrm{mp}^{\mathrm{g}}_{e,b,k_b}$

allows the BC node N_{k_b} to verify that the subfragment $\mathrm{sf}^{\mathrm{g}}_{e,b,k_b}$ is truly a piece of the validated $\mathrm{rh}^{\mathrm{g}}_{e,b}$.

Hash value agreement stage. This stage ensures that every node in the same BC will reach an agreement on a root hash (e.g., $\mathrm{rh}^{\mathrm{g}}_{e,b}$) corresponding to a valid local model fragment (e.g., $\mathbf{f}^{\mathrm{g}}_{e,b}$) even in the presence of Byzantine leader nodes. To this end, at the time point $t_e + \Delta_e^{\mathrm{Loc}} + \Delta_e^{\mathrm{C2B}} + \Delta_e^{\mathrm{C1}} + \Delta_e^{\mathrm{C2}} + \Delta_e^{\mathrm{C3}}$, a Byzantine tolerant consensus protocol, e.g., PBFT [288], is initiated by the leader node L_b who sends the message $\{e, \mathrm{sf}^{\mathrm{g}}_{e,b,k_b}, RA^{\mathrm{A}}_{e,L_b}, \mathrm{rh}^{\mathrm{g}}_{e,b}, \mathrm{mp}^{\mathrm{g}}_{e,b,k_b}\}$ to the BC node N_{k_b} $(\forall k_b \in [K_b], \forall b \in [B])$. The total time cost of this stage denotes Δ_e^{C4}.

Security analysis. The consensus protocol, combined with the remote attestation RA^{A}_{e,L_b} and the Merkle proofs, is used to detect Byzantine leader nodes and re-select a leader node. Once an honest leader node is selected, the agreement on a valid root hash can be reached among all honest BC nodes.

7.5 Simulation

In this section, the latency metrics of the CTP-FL system in terms of computation, communication and the Byzantine tolerant aggregation algorithm [263] are evaluated thoroughly.

7.5.1 Setup

Computation latency: All cryptographic computational performance benchmarks were executed on a laptop equipped with an Apple M2 processor and 16 GB of memory. To provide a realistic and reproducible measure of the overheads introduced by the two proposed protocols, we benchmarked each core cryptographic component using specialised libraries. Specifically, performance was measured as follows: the klauspost library [289] for erasure coding EC; a custom Rust implementation for the position-

binding polynomial commitment PBPC and remote attestation schemes; the PyCryptodome library [259] for message encryption/decryption; and the PyMerkle library [290] for Merkle proofs. The resulting fine-grained measurements were then used to calculate the total computational delays for each phase of the protocol, including Δ_e^{Loc1}, Δ_e^{Loc2}, Δ_e^{C1}, Δ_e^{C2}, Δ_e^{C3} and Δ_e^{C4}.

Communication latency: Network performance was simulated using the established fluid bandwidth-sharing model [291]. The model considers C clients, each with a local model of size S (MB), and a network of $B \times K$ blockchain nodes. Each local model is transformed into $B \times K$ subfragments and each subfragment is with the size of $\kappa S/(B \times K)$ MB (κ=1.5 is the EC parameter). We define the per-client uplink/downlink bandwidths as $b^{\mathrm{C}\uparrow}/b^{\mathrm{C}\downarrow}$ and per-BC node uplink/downlink bandwidths as $b^{\mathrm{B}\uparrow}/b^{\mathrm{B}\downarrow}$. To align with typical real-world conditions, $b^{\mathrm{C}\uparrow}/b^{\mathrm{B}\uparrow}$ and $b^{\mathrm{C}\downarrow}/b^{\mathrm{B}\downarrow}$ were set to 25 MB/s and 125 MB/s, respectively. The communication latency for every communication stage (e.g., Δ^{C2B} of the C2B stage) was computed via Eq. (7.18), with Δ^{P} representing propagation delay:

$$\Delta^{\mathrm{C2B}} = \min\left(b^{\mathrm{C}\uparrow}/(B \times K), b^{\mathrm{B}\downarrow}/C\right) + \Delta^{\mathrm{P}} \tag{7.18}$$

Local model training: The power load forecasting model employed is a Temporal Convolutional Network featuring residual dilated blocks and causal padding. Hyperparameters were set as follows: a hidden size of 64, a depth of 8 levels, a kernel size of 3, and a dropout probability of 0.1. The entire training process was implemented in Python and executed on a computer with an Intel Core i7-12700F CPU and an NVIDIA GeForce RTX 3090Ti GPU. Each client utilised stochastic gradient descent with a learning rate of 0.001 for local training.

Our evaluation employs the Australian Smart Grid, Smart City customer trial dataset [131], which contains energy consumption data from over 10,000 households in New South Wales. The raw data, recorded at a 30-minute resolution, was aggregated into hourly intervals by averaging every two consecutive samples. For our experiments, the period from 1 January to 31 March 2013 served as the training window, with the subsequent month (1 April to 30 April 2013) used for validation. Prior to training, all

load values were normalised to the range [0, 1]. Based on the above settings, the average time cost, i.e., Δ_e^{LT}, for 100MB, 200MB, 400MB, 600MB and 800MB sized models, was tested as 78.32s, 91.94s, 162.93s, 237.42s, and 303.88s, respectively.

The PLF input feature used in this evaluation is each customer's historical load value. The Australian Smart Grid, Smart City customer trial dataset [131] provides 30-minute energy-consumption records. These records are aggregated into hourly load values and then normalised to [0, 1]. This evaluation uses historical load only; weather variables and other exogenous features are outside the evaluated forecasting model and may change forecasting accuracy. They would not change the communication object analysed in this chapter, which is the transmitted local model or model update. Thus, *S* denotes the serialised size of the transmitted local model or update. The values *S*=100, 200, 400, 600 and 800 MB stress-test CTP-FL from moderate to large transmitted models or updates. For small models with few parameters and few clients, conventional FedAvg may be sufficient, but beyond the chapter's scope. The proposed architecture is intended for the larger setting described in Section 7.1.1, where larger model/update transmission becomes a material communication cost.

Attack and defence model: To consider the attack 2 mentioned in the security analysis of Section 7.3.1, the untargeted model poisoning attack [263], which deliberately biases the model convergence direction, is considered. The Byzantine-tolerant trimmed mean aggregation (TMA) algorithm [263] is applied to counter this untargeted attack:

$$\mathbf{f}_{e,b}^{\mathrm{g}}(d) = \frac{1}{C-2\beta} \sum_{c \in \boldsymbol{D}_{e,d}} \mathbf{f}_{e,c,b}^{\mathrm{l}}(d) \tag{7.19}$$

$$\mathbf{f}_{e,b}^{\mathrm{g}} = \left\{ \mathbf{f}_{e,b}^{\mathrm{g}}(1), \dots, \mathbf{f}_{e,b}^{\mathrm{g}}(d), \dots, \mathbf{f}_{e,b}^{\mathrm{g}}(D) \right\} \tag{7.20}$$

where D is the dimension number of the global model fragment $\mathbf{f}_{e,b}^{\mathrm{g}}$, $\mathbf{f}_{e,c,b}^{\mathrm{l}}(d)$ are the values of the local model fragment $\mathbf{f}_{e,c,b}^{\mathrm{l}}$ at the d^{th} dimension ($d \in [D]$, $c \in [C]$), $\mathbf{f}_{e,b}^{\mathrm{g}}(d)$ are the values at the d^{th} dimension of the global model fragment $\mathbf{f}_{e,b}^{\mathrm{g}}$, $\boldsymbol{D}_{e,d}$

is the set specifying the considered local model fragments whose parameters are at the d^{th} dimension in the e^{th} iteration, and β is a hyperparameter defining the number of outliers. Specifically, a total of $C - 2\beta$ client devices' indices are selected from $\boldsymbol{D}_{e,d}$, and the local model fragments corresponding to these indices are the remaining ones after removing the highest and lowest β values from $\{\mathbf{f}^{\mathrm{l}}_{e,1,b}(d), \ldots, \mathbf{f}^{\mathrm{l}}_{e,c,b}(d), \ldots, \mathbf{f}^{\mathrm{l}}_{e,C,b}(d)\}$ at the d^{th} dimension.

7.5.2 The Overall Latency Analysis of the Proposed System

Our primary finding is that the CTP-FL system successfully reduces the dominant time cost from communication in each FL model training iteration. As evidenced in Figure 7-3, the local training time Δ_e^{LT} consistently constitutes the majority of the total iteration time (denoted as Δ^{Tol}) across all tested scenarios, where the BC number *B*=[50, 75, 100], both client number (*C*) and node number per BC (*K*) are ranged from 100 to 200. Moreover, as the model size *S* increases, the distance between Δ_e^{LT} and $\Delta_e^{\mathrm{CommTol}}$ (or $\Delta_e^{\mathrm{O-Tol}}$) becomes even larger, where the total communication cost is $\Delta_e^{\mathrm{CommTol}} = \Delta_e^{\mathrm{C2B}} + \Delta_e^{\mathrm{B2C}} + \Delta_e^{\mathrm{C2C}}$ and $\Delta_e^{\mathrm{O-Tol}}$ is the other time cost of Δ^{Tol} except for $\Delta_e^{\mathrm{CommTol}}$ and Δ_e^{LT}. Furthermore, Figure 7-3(b) demonstrates the scalability of CTP-FL in communication time cost: $\Delta_e^{\mathrm{CommTol}}$ is nearly constant regardless of the number of clients (*C*), BCs (*B*), or nodes per BC (*K*). While the time cost $\Delta_e^{\mathrm{O-Tol}}$ scales with these parameters, the increase is marginal and remains negligible relative to Δ_e^{LT}, confirming that our introduced computational overhead is well-controlled.

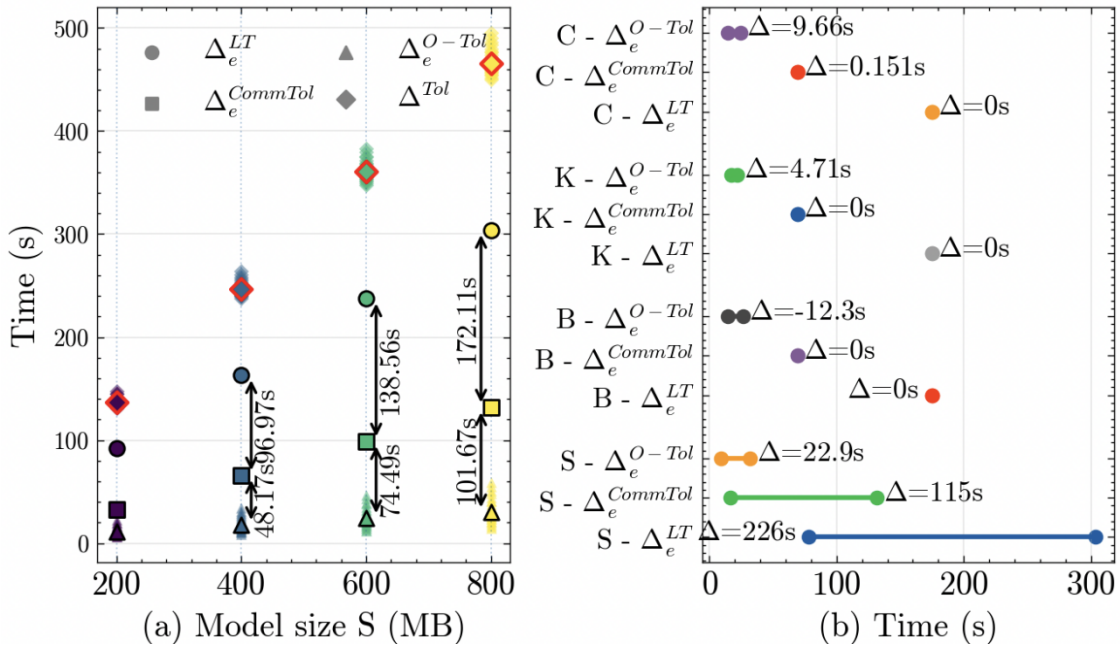

FIGURE 7-3. Evaluation on the time cost of the CTP-FL system.

7.5.3 The Latency Analysis of the Introduced Computation

The efficiency of CTP-FL is enabled by the cryptographic computations introduced in the C2B stage, and we now dissect this introduced computational overhead to rigorously validate its practicality. Figure 7-4 presents this breakdown by testing the two procedures' time costs, i.e., Δ_e^{LME} and Δ_e^{LFC}, detailed in Figure 7-2. Δ_e^{LME} consists of the fragment encryption cost Δ^{encr} and the EC encoding cost Δ^{enco}. It can be seen that although increasing S introduces a larger Δ^{encr}, increasing B can mitigate this side-effect. This is because increasing B reduces the size of a local model fragment (S/B), thereby reducing the encryption time cost Δ^{encr}. Figure 7-4(b) shows that although the model size (S), the number of BCs (B), and nodes per BC (K) can influence the value of Δ^{enco}, they are very small, such as around 0.3s when $S = 800$ and $K = 200$. This is because the EC encoding algorithm is computationally efficient and can be executed in parallel (see equation of Eq. (7.2)).

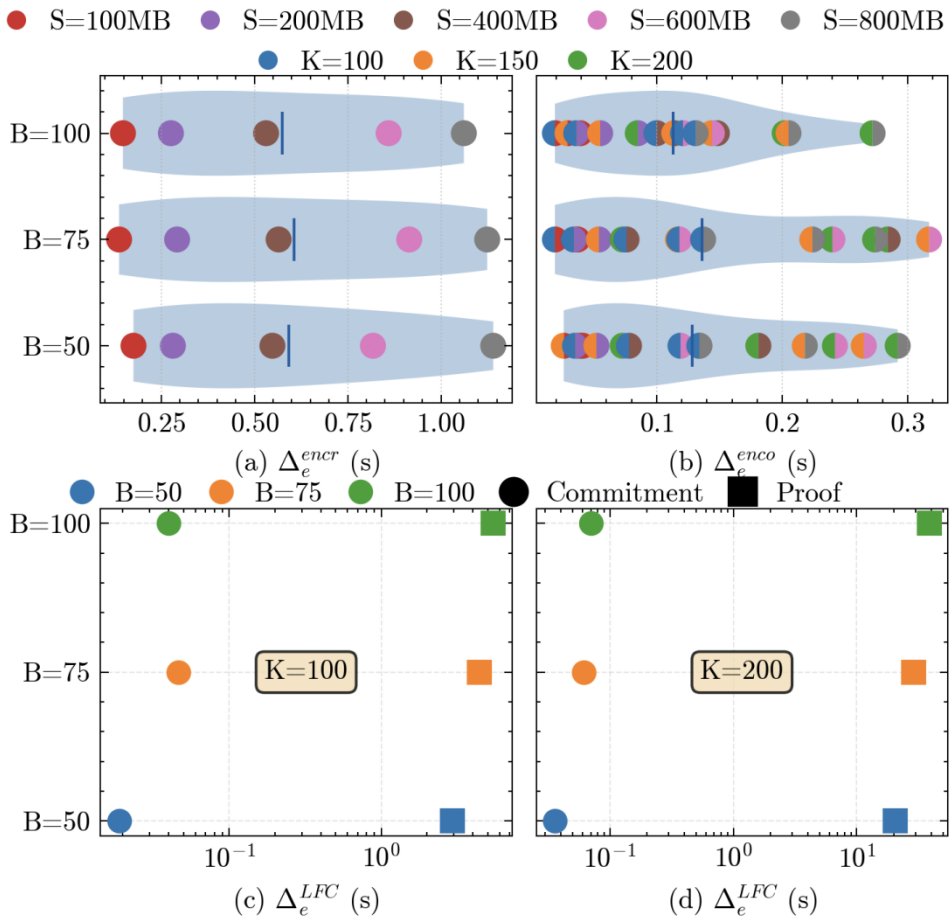

FIGURE 7-4. Evaluation on the time cost of Δ_e^{LME} and Δ_e^{LFC}.

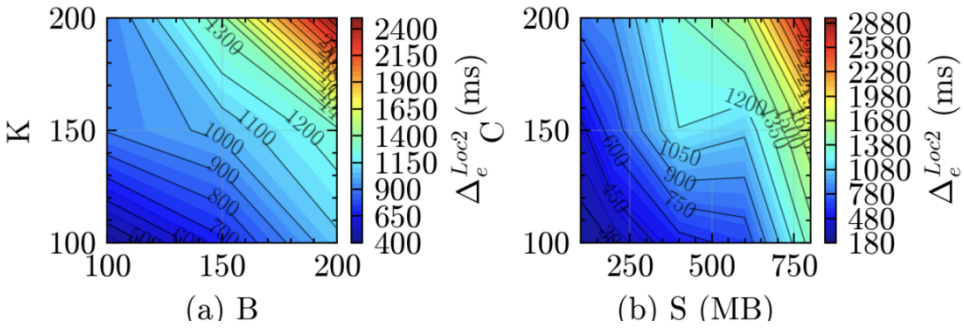

FIGURE 7-5. Evaluation on the time cost of Δ_e^{Loc2}.

The most significant source of overhead is the proof generation within the commitment procedure (Δ_e^{LFC}), driven by computationally intensive Multi-Scalar Multiplication (MSM) operations. However, this cost remains a small and acceptable fraction of the overall process. As shown in Figure 7-4(c), Figure 7-4(d) and Figure 7-3(a), even in the most demanding configurations (B=100, K=200), the proof generation time is still at least 8 times smaller than the local model training time Δ_e^{LT}.

This result indicates the controllability of the introduced computation overheads at the C2B stage.

Having established the practicality of the C2B stage's computational overheads, we now verify the computational time cost in the C2C stage. As shown in Figure 7-5, the double EC decoding performed by clients during the C2C stage introduces only a marginal delay Δ_e^{Loc2}. Despite scaling with the four parameters (node number per BC *K*, BC number *B*, model size *S* and client number *C*), this decoding time remains under 3 seconds across all scenarios – a trivial proportion of the local model training time Δ_e^{LT}. The reason lies in the high computation efficiency and concurrency nature of EC decoding operations. Notably, since the double EC encoding operations designed for reducing the B2C stage's communication cost are executed within the dual consensus protocol, it will be considered during measuring Δ_e^{C3} and Δ_e^{C4}.

7.5.4 The Latency Analysis of the Dual Consensus Protocol

Figure 7-6 illustrates the latency breakdown across the dual consensus protocol's four stages under various configurations. We observe two critical scalability features. First, the protocol scales well with the number of clients (*C*), as Δ_e^{C2} and Δ_e^{C4} is without growing when *C* increases. While the initial local message processing stage (Δ_e^{C1}) technically grows with *C*, its latency remains in the millisecond range. This is because this stage only involves the generation and broadcast of small pre-vectors. Second, and more importantly, the total consensus time demonstrates a favourable inverse relationship with the number of BCs (*B*). Increasing *B* significantly reduces overall latency because each BC leader is only responsible for processing smaller model fragments (*S/B*).

Figure 7-6 identifies the third stage's time cost Δ_e^{C3} as the primary contributor to the protocol's latency. We therefore dissect this phase further to scrutinise its internal composition and confirm its manageable nature (Figure 7-7). The vast majority of the Δ_e^{C3} cost is dominated by the execution of the Byzantine-tolerant aggregation algorithm

TMA (Δ^{agg}), while other cryptographic operations like Merkle proof generation (Δ_e^{mt} and Δ_e^{hash}) and EC encoding (Δ_e^{enc}) contribute negligible overhead. Crucially, even this dominant Δ^{agg} component exhibits excellent scalability: as the number of BC (B) increases, the size of the fragment aggregated by each leader decreases, leading to a reduction in Δ^{agg}. The results indicate the ability of the CTP-FL system for adopting computationally heavy, yet more robust model aggregation algorithms beyond TMA by properly setting system parameters.

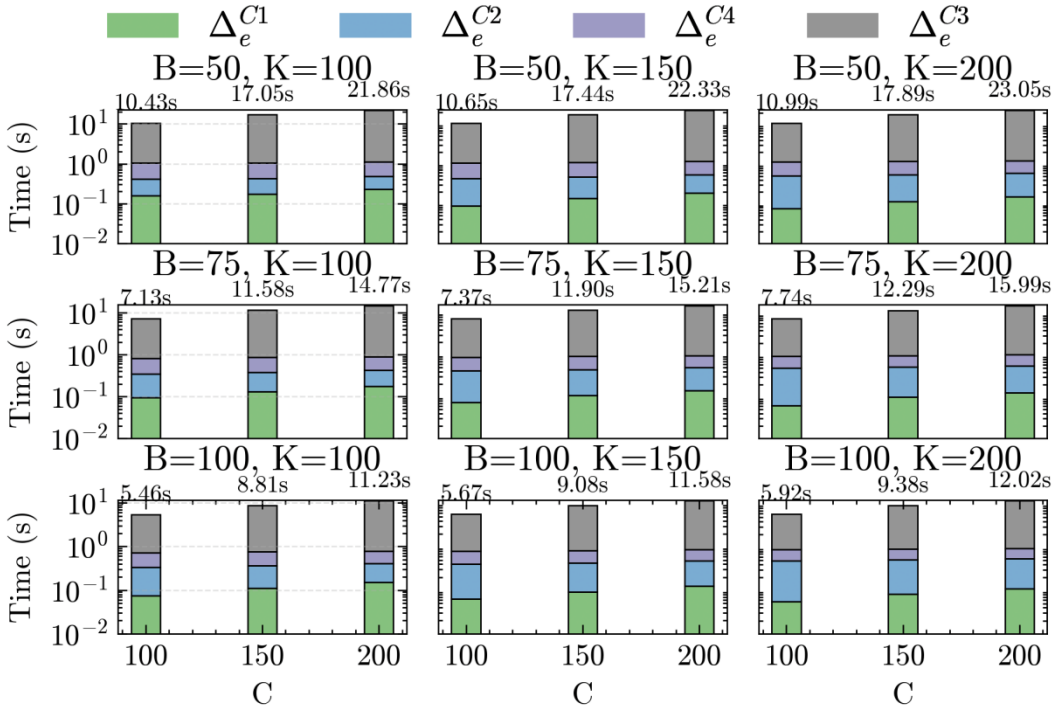

FIGURE 7-6. Evaluation on the time cost of the dual consensus protocol.

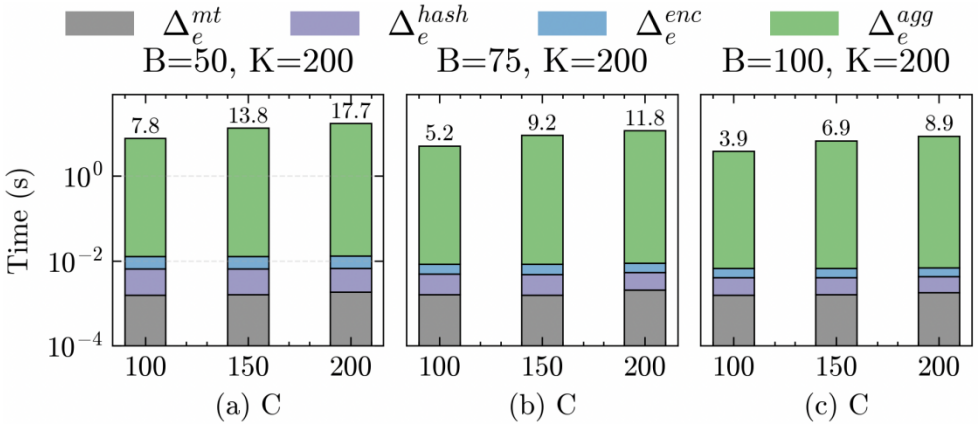

FIGURE 7-7. Evaluation on the time cost of Δ_e^{C3}.

7.5.5 The Latency Analysis of the Three Communication Stages

Having established the practicality of the system's computational and consensus overhead $\Delta_e^{\mathrm{O-Tol}}$, a breakdown of the total communication cost ($\Delta_e^{\mathrm{CommTol}}$) further reveals the paradigm's scalability with respect to client number (C) and model size (S) (Figure 7-8). While all stages scale linearly with S, the protocol exhibits exceptional scalability with C: both Δ_e^{C2B} and Δ_e^{C2C} are nearly constant, while Δ_e^{B2C} decreases as more clients join. This phenomenon is a direct result of the double EC strategy, which evenly distributes a part of BC nodes' data uploading overheads to all clients. Under this design, clients' uploading bandwidth is injected into the system, thereby improving and scaling the communication efficiency of this stage. Although the latency of the C2B and C2C stages is significant, it is inevitable since each client in each FL training iteration must upload its full local model for aggregation and download a full global model for next round's local model training. Overall, with the introduced computation overheads and the C2C stage, even in our worst-case test (S=800MB), $\Delta_e^{\mathrm{CommTol}}$ is only 132 seconds – approximately 2.3 times smaller than the local model training time Δ_e^{LT}, which confirms that communication is no longer the bottleneck.

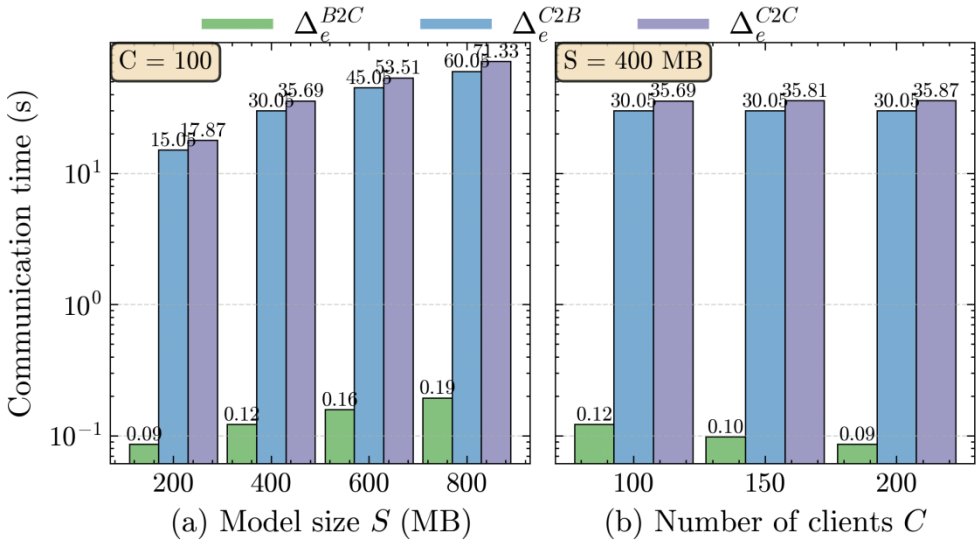

FIGURE 7-8. Evaluation on the time cost of the three-stage communication.

7.5.6 Latency Comparison

After illustrating the internal details of the CTP-FL system, we now benchmark its performance against the two FL paradigms: a centralised FL [263] and a BC-based FL [270] (BC-FL). To align with the real-world setting, we provisioned the aggregators in the baseline paradigms with a superior uplink bandwidth (125 MB/s), while keeping CTP-FL's parameters unchanged. The results presented in Figure 7-9 shows that CTP-FL is the only paradigm where its system's latency is dominated by local training time (Δ_e^{LT}), whereas both centralised and BC-FL paradigms are crippled by their high communication costs $\Delta_e^{\mathrm{CommTol}}$. Our paradigm reduces communication latency by at least 14.48x compared to centralised FL and a remarkable 20.26x compared to BC-FL.

More significantly, Figure 7-9 also reveals that the time cost $\Delta_e^{\mathrm{O-Tol}}$ in CTP-FL is even lower than those of the two baseline paradigms. This advantage is a direct consequence of our model partitioning design, which enables lightweight, parallelised aggregation of small fragments rather than full local models. Even in the most demanding scenario (S=800MB, C=200), the $\Delta_e^{\mathrm{O-Tol}}$ of CTP-FL is a mere 31.96 seconds in the CTP-FL system. The above results collectively confirm that the CTP-FL paradigm is primed for scalability even in larger model training scenarios.

The BC-FL baseline in Figure 7-9 represents a single-BC architecture. In this baseline, the BC leader and consensus protocol handle full local models. CTP-FL instead distributes one full model across B independent BCs. Each BC handles one model fragment of size approximately S/B, and its consensus and leader-side aggregation are executed only over that fragment. The B BCs operate independently during this phase, so their latency is not accumulated as B increases and no cross-chain protocol is invoked. Figure 7-9 includes cryptographic computations, erasure coding (EC), remote attestations, Merkle proofs and intra-BC consensus costs. Under this evaluation metric, CTP-FL reduces communication latency by 20.26x compared with BC-FL. The reduction comes from lowering the communication overhead of each BC without adding inter-BC communication or cross-chain verification steps.

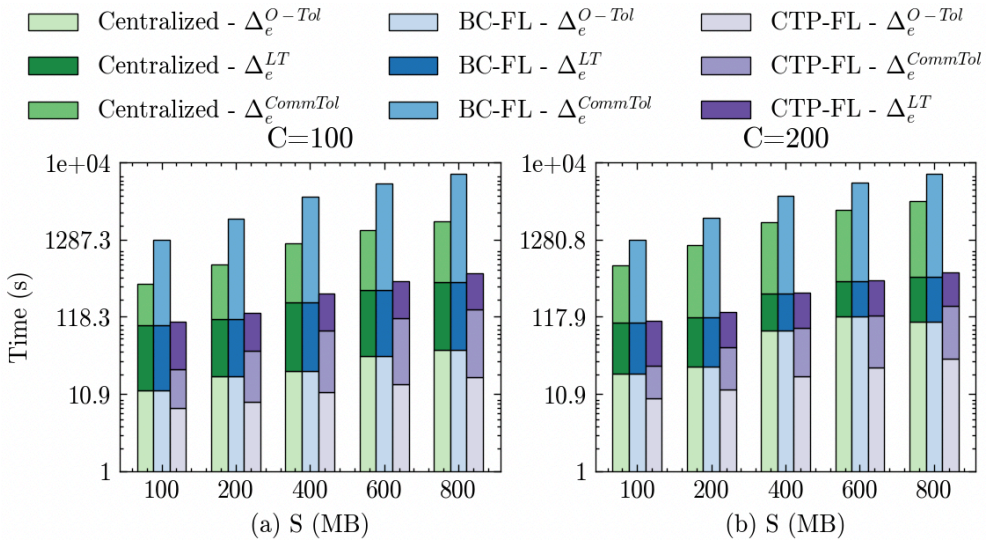

FIGURE 7-9. The overall time cost comparison among the three paradigms.

7.5.7 Byzantine Tolerance Analysis cross a Complete FL Process

After attesting the superior time-efficiency of CTP-FL, we now validate its second core promise: robust convergence in a hostile environment saturated with Byzantine participants. To this end, we subjected the system to a multi-pronged attack scenario with up to ε=30% Byzantine participants at both the client and BC sides (Figure 7-10). As shown in the security analysis of Section 7.3.1, Byzantine clients launched dual attacks: submitting unrecoverable fragments (attack 1) and injecting poisoned models [263] (attack 2), while Byzantine BC nodes were free to perform any malicious actions.

According to Figure 7-10, the system completely solves attack 1, which is due to the PBPC scheme specified in Section 7.2.2. By adopting this scheme, the leaderless consensus protocol detailed in Section 7.4.1 can directly eliminate attack 1 and only select recoverable local model fragments. As for the client-launched attack 2, by adopting the Byzantine-tolerant TMA algorithm, the system effectively mitigates its side effects. As shown in Figure 7-10(b), the global model reliably converges after 140 iterations, even when ε=30%. This indicates the ability of the system to integrate more robust algorithms to defend against more subtle attacks [292].

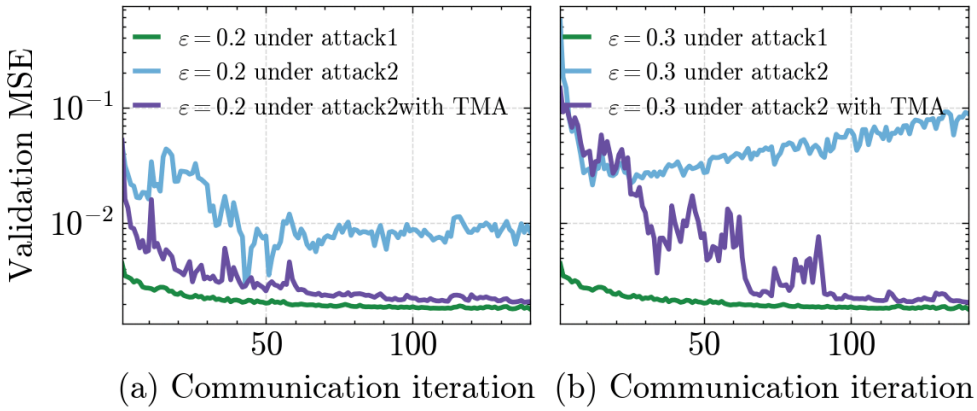

FIGURE 7-10. Evaluation on the convergence of CTP-FL under two attacks.

7.6 Conclusion

This chapter proposes CTP-FL, an FL paradigm for PLF that redistributes communication tasks across clients and BC nodes while supporting Byzantine-tolerant model aggregation. The CTP-FL paradigm introduces controllable computation overhead through two protocols: a message coding protocol for data recoverability and a dual consensus protocol for private and verifiable model-fragment aggregation. The simulation results show that, in the evaluated large-scale settings, CTP-FL reduces communication time compared with the centralised and BC-based FL paradigms while keeping the introduced computation overhead below the local training time. The results also show that the global model can converge under the considered Byzantine clients and Byzantine BC nodes. These findings support the feasibility of CTP-FL for secure and time-efficient FL-based power load forecasting under the assumptions and parameter ranges evaluated in this chapter.

Several limitations remain. First, the position-binding polynomial commitment scheme introduces client-side computation overhead, which may be demanding for resource-constrained clients who have less computational resources. Second, the current leader-based protocol assigns model-fragment aggregation to the BC leader node, so leader-side computation remains a design constraint. Third, CTP-FL redistributes part of the communication load to clients, especially in the client-to-client

(C2C) stage. It therefore relies on sufficient per-client uplink bandwidth. Clients with very limited uplink bandwidth may slow the C2C dissemination stage or require fallback handling. These limitations may restrict CTP-FL when training very large models, such as gigabyte-scale models. Future research should address them through more efficient commitment strategies, aggregation mechanisms that do not assign aggregation only to the BC leader and bandwidth-aware C2C dissemination for clients with weak uplinks.

CHAPTER 8
Conclusions and Future Directions

8.1 Conclusions

Distributed Energy Resources (DERs)-empowered energy systems have necessitated a robust, scalable, and privacy-preserving information infrastructure to underpin the increasingly sophisticated demand-side energy applications. The primary objective of the thesis is to construct a highly performant, robust foundation of the blockchain-based information infrastructure. Subsequently, the thesis demonstrates the efficacy of the infrastructure through two critical demand-side energy applications: Peer-to-Peer (P2P) energy trading and Power Load Forecasting (PLF).

This work yielded five key, demonstrable contributions that collectively advance the state-of-the-art in cyber-layer energy systems:

1. **Enhanced Scalability and Efficiency**: We introduced a novel dual-blockchain architecture to achieve vertical scalability by adding a computation layer on top of conventional blockchains. An Improved Optimistic Rollup (IOR) mechanism that significantly reduces the challenge periods common in traditional optimistic rollups (OR) is designed to ensure the security and robustness of the computation layer. The dual-blockchain system with the IOR mechanism significantly improve blockchain's throughput without sacrificing its security.
2. **Horizontal Security and Resilience**: We designed the protocol π^{τ} combined with a Transaction Batch Generation (TBG) protocol for Leaderless Byzantine Fault Tolerance (LBFT) consensus, substantially improving horizontal scalability and system resilience against three identified Byzantine attacks.
3. **Cross-Chain Interoperability**: The Blockchain-of-Blockchains (BoB) architecture was proposed, complete with the Cross-Chain Token Exchange (CCTE) and Cross-Chain Data Interoperability (CCDI) protocols, establishing a framework

that allows multiple independent blockchain platforms to securely interact and exchange data and assets with less latency and high throughput compared to the state-of-the-art cross-chain protocols.

4. **Privacy-Preserving, Time Efficient and Byzantine Tolerant P2P Market Clearing**: Combined with the advanced blockchain system, a highly efficient Trusted Execution Environment (TEE)-enabled distributed optimisation framework was developed. This framework, integrating the proposed D-TASK protocol and the novel TEAR-DO mechanism into the blockchain infrastructure, ensures time-efficient, Byzantine-tolerant, and energy data privacy preservation for distributed optimisation (DO)-based market clearing in P2P energy trading.
5. **Secure and Low-Latency Load Forecasting**: We pioneered CTP-FL ("**C**ommunication Burden **T**ransfers to all **P**articipants"-based Federated Learning), a new blockchain-based federated learning paradigm that uses a unique message coding protocol to dismantle the central aggregator bottleneck, and uses a dual-consensus protocol to achieve strong Byzantine tolerance at both client and server sides. By integrating the two protocols into the advanced blockchain infrastructure, the CTP-FL framework achieves Byzantine tolerant, communication-efficient and (local model parameters) privacy preserving FL for power load forecasting.

In summary, the blockchain-based information infrastructure developed in this thesis provides a comprehensive, secure, and high-performance solution for the next generation of demand-side energy applications. By rigorously addressing the system-level information infrastructure challenges first, this research further provides validated protocols and mechanisms that are directly applicable and compatible to the blockchain-based information infrastructure. With the integrated protocols and mechanisms, the blockchain-based information infrastructure can resolve the critical challenges inherently in the focused two demand-side energy applications. This devised blockchain-based information infrastructure may be of independent interest for a wide range of applications beyond the two focused energy applications.

8.2 Discussion of Practical Application Scenarios

The protocols and frameworks developed in Chapters 3-7 can support additional demand-side energy systems. These systems often coordinate distributed energy resources (DERs) owned by different participants. Local energy communities and multi-microgrid systems coordinate DERs across households, buildings or microgrids. Virtual power plants (VPPs) and aggregators combine flexible DERs so that they can participate in markets or grid-support services [295]. Demand response programmes adjust flexible demand in response to price or grid signals [295]. These settings require auditable records, optimisation or forecasting while limiting disclosure of local information. Transaction volume, data privacy and Byzantine behaviour therefore create cyber-layer problems after endpoint validation [1, 18, 19, 295].

The Improved Optimistic Rollup (IOR)-based dual-blockchain system in Chapter 3 applies when many operational records must remain auditable on a primary BC while computational overhead to process these records is prohibitive. Examples include EV charging payments, shared battery usage records, local flexibility commitments and local P2P trades inside energy communities. All these records may experience further processing via corresponding algorithms for their specific goals (e.g., maximising social welfare or minimising grid's operation cost). The Blockchain-of-Blockchains-based multi-blockchain system (BoB-MBS) in Chapter 5 supports BCs maintained by different operators. The Cross-Chain Token Exchange (CCTE) and Cross-Chain Data Interoperability (CCDI) protocols support token exchange and selected state data (SD) item updates across those BCs. Relevant operators may include communities, aggregators, VPPs and market platforms. Examples include cross-region P2P trading, inter-community flexibility exchange and coupled electricity-carbon-tradable green certificate (TGC) markets [294, 296]. Inter-community flexibility exchange means trading demand, generation or storage adjustment capacity across local energy communities.

The trusted execution environment (TEE)-assisted distributed optimisation (DO)

framework in Chapter 6 applies to demand-side coordination tasks that can be formulated as DO problems. Similar DO structures appear in demand response scheduling, network-constrained EV charging, distributed battery dispatch and local congestion management [233, 241, 295]. In these tasks, prosumers or aggregators may submit encrypted constraints, bids or utility/cost parameters. The framework keeps plaintext data inside TEEs while TEAR-DO computes the clearing or coordination result.

Communication Burden Transfers to all Participants-based Federated Learning (CTP-FL) in Chapter 7 applies to forecasting tasks beyond aggregate PLF. Examples include behind-the-meter load forecasting, photovoltaic (PV) generation forecasting and EV charging demand prediction. Behind-the-meter forecasting concerns customer-side load or generation behind a meter. Flexibility availability forecasting estimates future demand, generation or storage adjustment capacity. The task should involve many clients, non-negligible model sizes and Byzantine participants.

These scenarios follow the scope defined in Section 2.1.1. The proposed blockchain protocols operate after device-generated data, market requests or model updates have been fetched by participants. At that stage, the remaining risks include privacy leakage, invalid transactions generated by participants, incorrect computation results, inconsistent cross-chain state and Byzantine learning updates.

8.3 Future Directions

The comprehensive infrastructure and applications developed here open several promising avenues for future research:

1. **Refinement of data storage cost of optimistic rollup mechanism specified in Chapter 3:** While the dual-blockchain architecture is functional, its on-chain data storage cost is significant, since every transaction needs to be recorded in layer-1 for data availability – a key requirement for settling potential layer-2 disputes. Thus, how to mitigate this storage cost without sacrificing the data availability property

is worth investigating.

2. **The throughput improvement of leaderless Byzantine fault tolerance (LBFT) protocols (see Chapter 4) under heavy networks:** Although the designed TBG protocol significantly improves the throughput and latency of state-of-the-art LBFT protocols, its effectiveness will be reduced when the assumption in Theorem 4.4 does not hold, which is mainly caused by the network congestion – tens of thousands transactions are sent to the network within a short time window. Therefore, designing countermeasures to support this assumption is our future research direction.
3. **The impractical deposit mechanism used in CCTE and the communication burden of CCDI (see Chapter 5):** The high-efficiency of CCTE directly derives from the deposit mechanism which requires traders to deposit substantial of tokens in the BoB in advance, which is impractical to the participants who have inefficient financial balances. Additionally, The communication overhead of the CCDI protocol is relatively high since a proof generated by the BoB needs to be repeatedly sent to the targeted blockchain for security purposes. Resolving the token-deposit and communication issues will significantly improve the practicality of the two protocols.
4. **Scenario-specific deployment in demand-side energy systems (see Chapter 6 and Chapter 7)**: The discussion above identifies demand-side settings where the protocols and frameworks in this thesis are useful under the market clearing and power load forecasting scenarios. However, each application scenario has different market intervals, participant roles, bandwidth conditions and operator-intervention requirements. Future work should instantiate the relevant protocols in other specific scenarios before evaluating their generality. Relevant settings include EV charging coordination, aggregator-led demand response, VPP operation, inter-community flexibility exchange and forecasting services [295, 296].

Bibliography

[1] C. Guo, F. Luo, Z. Cai, and Z. Y. Dong, "Integrated energy systems of data centers and smart grids: State-of-the-art and future opportunities," *Appl. Energy*, vol. 301, p. 117474, 2021.

[2] G. Dileep, "A survey on smart grid technologies and applications," *Renewable Energy*, vol. 146, pp. 2589–2625, 2020.

[3] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid – The new and improved power grid: A survey," *IEEE Commun. Surv. Tutorials*, vol. 14, no. 4, pp. 944–980, 2012.

[4] E. A. Soto, L. B. Bosman, E. Wollega, and W. D. Leon-Salas, "Peer-to-peer energy trading: A review of the literature," *Appl. Energy*, vol. 283, p.116268, 2021.

[5] C. Wan, J. Zhao, Y. Song, Z. Xu, J. Lin, and Z. Hu, "Photovoltaic and solar power forecasting for smart grid energy management," *CSEE J. Power Energy Syst*., vol. 1, no. 4, pp. 38–46, 2015.

[6] M. B. Mollah, J. Zhao, D. Niyato, K.-Y. Lam, X. Zhang, A. M. Y. M. Ghias, L. H. Koh, and L. Yang, "Blockchain for future smart grid: A comprehensive

survey," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 18–43, 2021.

[7] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018.

[8] T. Nakai, A. Sakurai, S. Hironaka, and K. Shudo, "The blockchain trilemma described by a formula," *in Proc. IEEE Int. Conf. Blockchain*, 2023, pp. 41–46.

[9] P. Zhuang, T. Zamir, and H. Liang, "Blockchain for cybersecurity in smart grid: A comprehensive survey," *IEEE Trans. Ind. Inform.*, vol. 17, no. 1, pp. 3–19, 2021.

[10] K. Park, J. Lee, A. K. Das, and Y. Park, "BPPS: Blockchain-enabled privacy-preserving scheme for demand-response management in smart grid environments," I*EEE Trans. Depend. Secure Comput.*, vol. 20, no. 2, pp. 1719–1729, 2023.

[11] Z. Guo, P. Pinson, Q. Wu, S. Chen, Q. Yang, and Z. Yang, "An asynchronous online negotiation mechanism for real-time peer-to-peer electricity markets," *IEEE Trans. Power Syst.*, vol. 37, no. 3, pp. 1868–1880, 2022.

[12] J. Lei, L. Wang, Q. Pei, W. Sun, X. Lin, and X. Liu, "PrivGrid: Privacy-preserving individual load forecasting service for smart grid," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 6856–6870, 2024.

[13] Y. Wang, Z. Su, N. Zhang, J. Chen, X. Sun, Z. Ye, and Z. Zhou, "SPDS: A secure and auditable private data sharing scheme for smart grid based on blockchain," *IEEE Trans. Ind. Inform.*, vol. 17, no. 11, pp. 7688–7699, 2021.

[14] M. Raikwar, D. Gligoroski, and K. Kralevska, "SoK of Used Cryptography in Blockchain," *IEEE Access*, vol. 7, pp. 148550–148575, 2019.

[15] C. Liu, H. Guo, M. Xu, S. Wang, D. Yu, J. Yu, and X. Cheng, "Extending on-chain trust to off-chain – trustworthy blockchain data collection using trusted execution environment (TEE)," *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3268–3280, 2022.

[16] S. Kayikci and T. M. Khoshgoftaar, "Blockchain meets machine learning: A

survey," *J. Big Data*, vol. 11, no. 1, p. 9, 2024.

[17] S. Aman, Y. Simmhan, and V. K. Prasanna, "Energy management systems: State of the art and emerging trends," *IEEE Commun. Mag*., vol. 51, no. 1, pp. 114–119, 2013.

[18] M. S. Hossain, N. A. Madlool, N. A. Rahim, J. Selvaraj, A. K. Pandey, and A. F. Khan, "Role of smart grid in renewable energy: An overview," *Renew. Sustain. Energy Rev*., vol. 60, pp. 1168–1184, 2016.

[19] P. Kumar, Y. Lin, G. Bai, A. Paverd, J. S. Dong, and A. Martin, "Smart grid metering networks: A survey on security, privacy and open research issues," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 3, pp. 2886–2927, 2019.

[20] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[21] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[22] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst*., vol. 107, pp. 841–853, 2020.

[23] A. Miglani, N. Kumar, V. Chamola, and S. Zeadally, "Blockchain for Internet of Energy management: Review, solutions, and challenges," *Comput. Commun*., vol. 151, pp. 395–418, 2020.

[24] M. Zhang, F. Eliassen, A. Taherkordi, H.-A. Jacobsen, H.-M. Chung, and Y. Zhang, "Demand-response games for peer-to-peer energy trading with the Hyperledger blockchain," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 1, pp. 19–31, 2022.

[25] T. Alladi, V. Chamola, J. J. P. C. Rodrigues, and S. A. Kozlov, "Blockchain in smart grids: A review on different use cases," *Sensors*, vol. 19, no. 22, Art. no. 4862, 2019.

[26] S. Wang, A. F. Taha, J. Wang, K. Kvaternik, and A. Hahn, "Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 49, no. 8, pp. 1612–1623,

2019.

[27] Bhawana, S. Kumar, R. S. Rathore, U. Dohare, O. Kaiwartya, J. Lloret, and N. Kumar, "BEET: Blockchain enabled energy trading for e-mobility oriented electric vehicles," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3018–3034, 2024.

[28] 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 Trans. Smart Grid*, vol. 10, no. 3, pp. 3162–3173, 2019.

[29] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos, "Designing blockchain-based access control protocol in IoT-enabled smart-grid system," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5744–5761, 2021.

[30] S. S. Uddin, et al., "Next-generation blockchain enabled smart grid: Conceptual framework, key technologies and industry practices review," *Energy AI*, vol. 12, Art. no. 100228, 2023.

[31] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93039–93054, 2022.

[32] S. Schulte, M. Sigwart, P. Frauenthaler, and M. Borkowski, "Towards blockchain interoperability," in *Proc. Int. Conf. Bus. Process Manag.*, 2019, pp. 3–10.

[33] N. Papadis and L. Tassiulas, "Blockchain-based payment channel networks: Challenges and recent advances," *IEEE Access*, vol. 8, pp. 227596–227609, 2020.

[34] L. D. Negka and G. P. Spathoulas, "Blockchain state channels: A state of the art," *IEEE Access*, vol. 9, pp. 160277–160298, 2021.

[35] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," white paper, 2017.

[36] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of Layer-two blockchain protocols," *J. Netw. Comput. Appl.*, vol. 209, art. no. 103539, 2023.

[37] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of

blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

[38] M. Saif, et al., "A survey on data availability in layer 2 blockchain rollups: Open challenges and future improvements," *Future Internet*, vol. 16, no. 9, art. no. 315, 2024.

[39] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 1353–1370.

[40] O. Patel, "Blockchain Layer 2 Rollups: Optimistic vs Zero Knowledge," *Int. J. Adv. Res. Eng. Technol*., vol. 14, no. 2, pp. 13–29, 2023.

[41] S. Motepalli, L. Freitas, and B. Livshits, "SoK: Decentralised sequencers for rollups," *arXiv* preprint arXiv:2310.03616, 2023.

[42] N. R. Pradhan, A. P. Singh, N. Kumar, M. M. Hassan, and D. S. Roy, "A flexible permission ascription (FPA)-based blockchain framework for peer-to-peer energy trading with performance evaluation," *IEEE Trans. Ind. Inform.*, vol. 18, no. 4, pp. 2465–2475, 2021.

[43] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 1–35, 2023.

[44] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer PBFT consensus for blockchain," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1146–1160, 2020.

[45] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus," *arXiv preprint arXiv*:1807.04938, 2018.

[46] M. M. Jalalzai, J. Niu, C. Feng, and F. Gai, "Fast-hotstuff: A fast and robust BFT protocol for blockchains," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 2478–2493, 2023.

[47] S. Liu, W. Xu, C. Shan, X. Yan, T. Xu, B. Wang, L. Fan, F. Deng, Y. Yan, and H. Zhang, "Flexible advancement in asynchronous BFT consensus," in *Proc. 29th Symp. Oper. Syst. Princ.*, 2023, pp. 264–280.

[48] L. Baird and A. Luykx, "The hashgraph protocol: Efficient asynchronous BFT

for high-throughput distributed ledgers," in *Proc. Int. Conf. Omni-layer Intell. Syst.*, 2020, pp. 1–7.

[49] P. Jovanovic, L. Kokoris-Kogias, B. Kumara, A. Sonnino, P. Tennage, and I. Zablotchi, "Mahi-Mahi: Low-latency asynchronous BFT DAG-based consensus," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2025, pp. 549–559.

[50] X. Dai, Z. Zhang, J. Xiao, J. Yue, X. Xie, and H. Jin, "GradedDAG: An asynchronous DAG-based BFT consensus with lower latency," in *Proc. IEEE Int. Symp. Rel. Distrib. Syst. (SRDS)*, 2023, pp. 107–117.

[51] A. Gągol, D. Leśniak, D. Straszak, and M. Świętek, "Aleph: Efficient atomic broadcast in asynchronous networks with Byzantine nodes," in *Proc. ACM Conf. Adv. Financial Technol.*, 2019, pp. 214–228.

[52] Z. An, M. Wang, D. Liu, T. Li, and Q. Lai, "Hydra: An efficient asynchronous DAG-based BFT protocol," in *Proc. Int. Conf. Sci. Cyber Secur.*, 2023, pp. 439–459.

[53] H. Guo, M. Xu, J. Zhang, C. Liu, R. Ranjan, D. Yu, and X. Cheng, "BFT-DSN: A Byzantine fault-tolerant decentralized storage network," *IEEE Trans. Comput.*, vol. 73, no. 5, pp. 1300–1312, 2024.

[54] C. Berger, S. Schwarz-Rüsch, A. Vogel, K. Bleeke, L. Jehl, H. P. Reiser, and R. Kapitza, "SoK: Scalability techniques for BFT consensus," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency*, 2023, pp. 1–18.

[55] S. Bonomi, A. Del Pozzo, M. Potop-Butucaru, and S. Tixeuil, "Approximate agreement under mobile Byzantine faults," *Theor. Comput. Sci*., vol. 758, pp. 17–29, Feb. 2019.

[56] Y. Shang, "Resilient consensus for multiagent networks with time-varying delay and mobile adversaries," *J. Phys. Complex*., vol. 5, no. 4, p. 045010, 2024.

[57] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing, or: How to cope with perpetual leakage," in *Proc. Int. Cryptology Conf.*, 1995, pp. 339–352.

[58] A. Zamyatin, et al., "SoK: Communication across distributed ledgers," in *Proc.*

Int. Conf. Financ. Cryptogr. Data Secur., 2021, pp. 3–36.

[59] T. Hardjono, A. Lipton, and A. Pentland, "Toward an interoperability architecture for blockchain autonomous systems," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1298–1309, 2020.

[60] A. Deshpande and M. Herlihy, "Privacy-preserving cross-chain atomic swaps," in *Proc. Int. Conf. Financ. Cryptogr. Data Secur.*, 2020, pp. 540–549.

[61] E. Clark, et al., "On HTLC-based protocols for multi-party cross-chain swaps," in *Proc. Int. Symp. Algorithms Comput.*, 2024, pp. 22:1–22:15.

[62] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019.

[63] G. Falazi, U. Breitenbücher, F. Leymann, and S. Schulte, "Cross-chain smart contract invocations: A systematic multi-vocal literature review," *ACM Comput. Surv.*, vol. 56, no. 6, pp. 1–38, 2024.

[64] K. Singh, V. J. Ribeiro, and S. Mandal, "Flexiswap: A cross-chain decentralized exchange protocol," in *Proc. Int. Conf. COMmun. Syst. NETw.*, 2025, pp. 730–738.

[65] S. Mazumdar, "Towards faster settlement in HTLC-based cross-chain atomic swaps," in *Proc. IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl.*, 2022, pp. 295–304.

[66] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," White Paper, 2016. [Online]. Available: https://polkadot.network

[67] C. Goes, E. Buchman, Ž. Milosevic, I. Khoffi, and C. Leung, "The Inter-Blockchain Communication (IBC) protocol: An overview," White Paper, 2020. [Online]. Available: https://ibcprotocol.org

[68] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–41, 2022.

[69] P. Han, Z. Yan, W. Ding, S. Fei, and Z. Wan, "A survey on cross-chain

technologies," *Distrib. Ledger Technol.: Res. Pract.*, vol. 2, no. 2, pp. 1–30, 2023.

[70] H. Yuan, S. Fei, and Z. Yan, "Technologies of blockchain interoperability: A survey," *Digit. Commun. Netw.*, vol. 11, no. 1, pp. 210–224, 2025.

[71] R. Belchior, P. Somogyvari, J. Pfannschmidt, A. Vasconcelos, and M. Correia, "Hephaestus: Modeling, analysis, and performance evaluation of cross-chain transactions," *IEEE Trans. Reliab.*, vol. 73, no. 2, pp. 1132–1146, 2024.

[72] M. Borkowski, M. Sigwart, P. Frauenthaler, T. Hukkinen, and S. Schulte, "DeXTT: Deterministic cross-blockchain token transfers," *IEEE Access*, vol. 7, pp. 111030–111042, 2019.

[73] C. G. Harris, "Cross-chain technologies: Challenges and opportunities for blockchain interoperability," in *Proc. IEEE Int. Conf. Omni-Layer Intell. Syst.*, 2023, pp. 1–6.

[74] T. Yu, F. Luo, G. Ranzi, and J. Wu, "Secure and efficient data interoperability protocols for multi-blockchains systems," *IEEE Trans. Inf. Forensics Security*, vol. 20, pp. 11401–11416, 2025.

[75] 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 Trans. Ind. Inform.*, vol. 18, no. 1, pp. 231–241, 2022.

[76] M. K. Thukral, "Emergence of blockchain-technology application in peer-to-peer electrical-energy trading: A review," *Clean Energy*, vol. 5, no. 1, pp. 104–123, 2021.

[77] S. Chen, Z. Shen, L. Zhang, Z. Yan, C. Li, N. Zhang, and J. Wu, "A trusted energy trading framework by marrying blockchain and optimisation," *Adv. Appl. Energy*, vol. 2, p. 100029, 2021.

[78] C. Shah, J. King, and R. W. Wies, "Distributed ADMM using private blockchain for power flow optimisation in distribution network with coupled and mixed-integer constraints," *IEEE Access*, vol. 9, pp. 46560–46572, 2021.

[79] E. Münsing, J. Mather, and S. Moura, “Blockchains for decentralized optimisation of energy resources in microgrid networks,” in *Proc. IEEE Conf. Control Technol. Appl.*, 2017, pp. 2164–2171.

[80] S. Mao, Y. Tang, Z. Dong, K. Meng, Z. Y. Dong, and F. Qian, “A privacy preserving distributed optimisation algorithm for economic dispatch over time-varying directed networks,” *IEEE Trans. Ind. Inform.*, vol. 17, no. 3, pp. 1689–1701, 2021.

[81] M. Sivianes, J. M. Maestre, A. Zafra-Cabeza, and C. Bordons, “Blockchain for energy trading in energy communities using stochastic and distributed model predictive control,” *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 5, pp. 2132–2145, 2023.

[82] B. Wang, L. Xu, and J. Wang, “A privacy-preserving trading strategy for blockchain-based P2P electricity transactions,” *Appl. Energy*, vol. 335, p. 120664, 2023.

[83] X. Chang, Y. Xu, H. Sun, and Q. Wu, “Privacy-preserving distributed energy transaction in active distribution networks,” *IEEE Trans. Power Syst.*, vol. 38, no. 4, pp. 3413–3426, 2023.

[84] N. Tian, Q. Guo, H. Sun, and X. Zhou, “Fully privacy-preserving distributed optimisation in power systems based on secret sharing,” *iEnergy*, vol. 1, no. 3, pp. 351–362, 2022.

[85] W. An, D. Ding, H. Dong, B. Shen, and L. Sun, “Privacy-preserving distributed optimisation for economic dispatch over balanced directed networks,” *IEEE Trans. Inf. Forensics Security*, vol. 20, pp. 1362–1373, 2025.

[86] H. Liu, S. Lei, J. Hu, and Y. Luo, “Privacy attack-resilient peer-to-peer energy trading over directed networks,” *IEEE Trans. Smart Grid*, vol. 16, no. 6, pp. 5673–5676, 2025.

[87] J. Guo, X. Ding, and W. Wu, “An architecture for distributed energies trading in Byzantine-based blockchains,” *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 2, pp. 1216–1230, 2022.

[88] J. Liu, Q. Long, R.-P. Liu, W. Liu, and Y. Hou, “Byzantine-resilient distributed P2P energy trading via spatial-temporal anomaly detection,” *IEEE Trans. Smart Grid*, vol. 16, no. 5, pp. 4045–4057, 2025.

[89] D. H. Nguyen, “A cooperative learning approach for decentralized peer-to-peer energy trading markets and its structural robustness against cyberattacks,” *IEEE Access*, vol. 9, pp. 148862–148872, 2021.

[90] D. J. Sebastian, U. Agrawal, A. Tamimi, and A. Hahn, “DER-TEE: Secure distributed energy resource operations through trusted execution environments,” *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6476–6486, 2019.

[91] X. Chang, Y. Xu, Q. Guo, H. Sun, and W. K. Chan, “A Byzantine-resilient distributed peer-to-peer energy management approach,” *IEEE Trans. Smart Grid*, vol. 14, no. 1, pp. 623–634, 2023.

[92] J. D. Fernández, S. P. Menci, C. M. Lee, A. Rieger, and G. Fridgen, “Privacy-preserving federated learning for residential short-term load forecasting,” *Appl. Energy*, vol. 326, art. no. 119915, 2022.

[93] C. Briggs, Z. Fan, and P. Andras, “Federated learning for short-term residential load forecasting,” *IEEE Open Access J. Power Energy*, vol. 9, pp. 573–583, 2022.

[94] A. Taïk and S. Cherkaoui, “Electrical load forecasting using edge computing and federated learning,” in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

[95] N. Gholizadeh and P. Musilek, “Federated learning with hyperparameter-based clustering for electrical load forecasting,” *Internet of Things*, vol. 17, art. no. 100470, 2022.

[96] M. A. Husnoo, A. Anwar, N. Hosseinzadeh, S. N. Islam, A. N. Mahmood, and R. Doss, “A secure federated learning framework for residential short term load forecasting,” *IEEE Trans. Smart Grid*, vol. 15, no. 2, pp. 2044–2055, 2024.

[97] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, “A survey on federated learning: challenges and applications,” *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 2, pp. 513–535, 2023.

[98] Z. Zhao, Y. Mao, Y. Liu, L. Song, Y. Ouyang, X. Chen, and W. Ding, "Towards efficient communications in federated learning: A contemporary survey," *J. Franklin Inst.*, vol. 360, no. 12, pp. 8669–8703, 2023.

[99] P. Liu, X. Xu, and W. Wang, "Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives," *Cybersecurity*, vol. 5, no. 1, art. no. 4, 2022.

[100] M. Antal, V. Mihailescu, T. Cioara, and I. Anghel, "Blockchain-based distributed federated learning in smart grid," *Mathematics*, vol. 10, no. 23, art. no. 4499, Nov. 2022.

[101] M. Wang, et al., "A federated LSTM network for load forecasting using multi-source data with homomorphic encryption," *AIMS Energy*, vol. 13, no. 2, pp. 265–289, 2025.

[102] Y. Parag and B. K. Sovacool, "Electricity market design for the prosumer era," *Nat. Energy*, vol. 1, no. 4, Art. no. 16032, 2016.

[103] W. Tushar, T. K. Saha, C. Yuen, D. Smith, and H. V. Poor, "Peer-to-peer trading in electricity networks: An overview," *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 3185–3200, 2020.

[104] F. Luo, Z. Y. Dong, G. Liang, J. Murata, and Z. Xu, "A distributed electricity trading system in active distribution networks based on multi-agent coalition and blockchain," *IEEE Trans. Power Syst.*, vol. 34, no. 5, pp. 4097–4108, 2019.

[105] J. Abdella, Z. Tari, A. Anwar, A. Mahmood, and F. Han, "An architecture and performance evaluation of blockchain-based peer-to-peer energy trading," *IEEE Trans. Smart Grid*, vol. 12, no. 4, pp. 3364–3378, 2021.

[106] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang, "Blockchain-enabled secure energy trading with verifiable fairness in Industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6564–6574, 2020.

[107] A. Dorri, F. Luo, S. S. Karumba, S. Kanhere, R. Jurdak, and Z. Y. Dong, "Temporary immutability: A removable blockchain solution for prosumer-side energy trading," *J. Netw. Comput. Appl.*, vol. 180, Art. no. 103018, 2021.

[108] M. R. Hamouda, M. E. Nassar, and M. M. A. Salama, "A novel energy trading

framework using adapted blockchain technology," *IEEE Trans. Smart Grid*, vol. 12, no. 3, pp. 2165–2175, 2021.

[109] A. Dorri, F. Luo, S. S. Kanhere, R. Jurdak, and Z. Y. Dong, "SPB: A secure private blockchain-based solution for distributed energy trading," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 120–126, 2019.

[110] M. K. AlAshery, et al., "A blockchain-enabled multi-settlement quasi-ideal peer-to-peer trading framework," *IEEE Trans. Smart Grid*, vol. 12, no. 1, pp. 885–896, 2021.

[111] J. Guo, X. Ding, and W. Wu, "A blockchain-enabled ecosystem for distributed electricity trading in smart city," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 2040–2050, 2021.

[112] 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," *Appl. Energy*, vol. 295, Art. no. 117056, 2021.

[113] E. Mengelkamp, J. Gärttner, K. Rock, S. Kessler, L. Orsini, and C. Weinhardt, "Designing microgrid energy markets: A case study: The Brooklyn Microgrid," *Appl. Energy*, vol. 210, pp. 870–880, 2018.

[114] Power Ledger, "Where power meets blockchain," Power Ledger, Perth, Australia. Accessed: Dec. 2025. [Online]. Available: https://powerledger.io

[115] M. Aloqaily, A. Boukerche, O. Bouachir, F. Khalid, and S. Jangsher, "An energy trade framework using smart contracts: Overview and challenges," *IEEE Netw.*, vol. 34, no. 4, pp. 119–125, 2020.

[116] V. Buterin, "Ethereum: Platform review," in *Opportunities and Challenges for Private and Consortium Blockchains*, pp. 1–45, 2016.

[117] C. Egger, P. Moreno-Sanchez, and M. Maffei, "Atomic multi-channel updates with constant collateral in Bitcoin-compatible payment-channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 801–815.

[118] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable off-chain instant payments," white paper, 2016. [Online]. Available:

https://lightning.network/lightning-network-paper.pdf

[119] Raiden Network, “Raiden Network,” Accessed: Dec. 2025. [Online]. Available: https://raiden.network

[120] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo, “Sidechain technologies in blockchain networks: An examination and state-of-the-art review,” *J. Netw. Comput. Appl.*, vol. 149, Art. no. 102471, 2020.

[121] J. Adler and M. Quintyne-Collins, “Building scalable decentralized payment systems,” *arXiv* preprint arXiv:1904.06441, 2019.

[122] C. Sguanci, R. Spatafora, and A. M. Vergani, “Layer 2 blockchain scaling: A survey,” *arXiv* preprint arXiv:2107.10881, 2021.

[123] Ethereum Foundation, “Optimistic rollups,” Ethereum developer documentation. Accessed: Dec. 2025. [Online]. Available: https://ethereum.org/developers/docs/scaling/optimistic-rollups/

[124] Ethereum Foundation, “Zero-knowledge rollups,” Ethereum developer documentation. Accessed: Dec. 2025. [Online]. Available: https://ethereum.org/developers/docs/scaling/zk-rollups/

[125] Optimism Collective, “Rollup protocol overview,” Optimism documentation. Accessed: Dec. 2025. [Online]. Available: https://docs.optimism.io/op-stack/protocol/overview

[126] A. Gluchowski, “Introducing zkSync: the missing link to mass adoption of Ethereum,” Matter Labs Blog, 2020. Accessed: Dec. 2025. [Online]. Available: https://medium.com/matter-labs/introducing-zk-sync-the-missing-link-to-mass-adoption-of-ethereum-14c9cea83f58

[127] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, “Scalable, transparent, and post-quantum secure computational integrity,” *Cryptology ePrint Archive*, Rep. 2018/046, 2018.

[128] O. Goldreich and Y. Oren, “Definitions and properties of zero-knowledge proof systems,” *J. Cryptol.*, vol. 7, no. 1, pp. 1–32, 1994.

[129] FISCO BCOS, “FISCO BCOS: Open-source consortium blockchain platform,”

Financial Blockchain Shenzhen Consortium. Accessed: Dec. 2025. [Online]. Available: https://www.fisco-bcos.org

[130] Alibaba Cloud, “Alibaba Cloud official website,” Accessed: Dec. 2025. [Online]. Available: https://www.aliyun.com

[131] Australian Government, “Smart Grid, Smart City: National energy efficiency initiative – Program overview,” Dept. Resources, Energy and Tourism, Canberra, ACT, Australia, 2014. Accessed: Dec. 2025. [Online]. Available: https://www.industry.gov.au/

[132] Z. Zhao, F. Luo, C. Zhang, and G. Ranzi, “A social relationship preference aware peer-to-peer energy market for urban energy prosumers and consumers,” *IET Renew. Power Gener.*, vol. 16, no. 4, pp. 688–699, 2022.

[133] M. Iansiti and K. R. Lakhani, “The truth about blockchain,” *Harvard Bus. Rev.*, vol. 95, no. 1, pp. 118–127, Jan./Feb. 2017.

[134] S. Gupta, S. Rahnama, S. Pandey, N. Crooks, and M. Sadoghi, “Dissecting BFT consensus: In trusted components we trust!” in *Proc. 18th Eur. Conf. Comput. Syst.*, 2023, pp. 521–539.

[135] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, “The next 700 BFT protocols,” *ACM Trans. Comput. Syst.*, vol. 32, no. 4, pp. 1–45, 2015.

[136] G. Zhang, F. Pan, et al., “PrestigeBFT: Revolutionizing view changes in BFT consensus algorithms with reputation mechanisms,” in *Proc. IEEE Int. Conf. Data Eng*., 2024, pp. 1930–1943.

[137] M. Yin, D. Malkhi, et al., “HotStuff: BFT consensus with linearity and responsiveness,” in *Proc. ACM Symp. Principles Distrib. Comput*., 2019, pp. 347–356.

[138] X. Sui, Q. Liu, S. Duan, and H. Zhang, “Pike: Two-Phase BFT with Linearity and Flexible View Change,” *IEEE Trans. Comput.*, vol. 74, no. 8, pp. 27722784, 2025.

[139] F. Gai, J. Niu, et al., “Scaling blockchain consensus via a robust shared mempool,” in *Proc. IEEE Int. Conf. Data Eng*., 2023, pp. 530–543.

[140] C. Copeland and H. Zhong, "Tangaroa: A Byzantine fault tolerant Raft," *Stanford Univ*., 2016.

[141] A. Miller, Y. Xia, et al., "The Honey Badger of BFT protocols," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2016, pp. 31–42.

[142] Y. Gao, Y. Lu, et al., "Dumbo-ng: Fast asynchronous BFT consensus with throughput-oblivious latency," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2022, pp. 1187–1201.

[143] X. Dai, C. Ding, et al., "Falcon: Advancing asynchronous BFT consensus for lower latency and enhanced throughput," *arXiv preprint arXiv:2504.12766*, 2025.

[144] H. Cheng, Y. Lu, et al., "JUMBO: Fully asynchronous BFT consensus made truly scalable," *arXiv preprint arXiv:2403.11238*, 2024.

[145] Y. Manevich, "Arma: Byzantine fault tolerant consensus with linear scalability," *arXiv preprint arXiv:2312.13777*, 2023.

[146] C. Stathakopoulou, T. David, and M. Vukolic, "Mir-BFT: High-throughput BFT for blockchains," *arXiv preprint arXiv:1906.05552*, 2019.

[147] Z. Avarikioti, et al., "Fnf-BFT: Exploring performance limits of BFT protocols," *arXiv preprint arXiv:2009.02235*, 2020.

[148] D. S. Antunes, A. N. Oliveira, et al., "Alea-BFT: Practical asynchronous Byzantine fault tolerance," in *Proc. USENIX Symp. Netw. Syst. Des. Implement*., 2024, pp. 313–328.

[149] S. Duan, et al., "BEAT: Asynchronous BFT made practical," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2018, pp. 2028–2041.

[150] R. Canetti, et al., "Fast asynchronous Byzantine agreement with optimal resilience," in *Proc. ACM Symp. Theory Comput*., 1993, pp. 42–51.

[151] H. Zhang and S. Duan, "PACE: Fully parallelizable BFT from reproposable Byzantine agreement," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2022, pp. 3151–3164.

[152] A. Spiegelman, N. Giridharan, et al., "Bullshark: DAG BFT protocols made

practical," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2022, pp. 2705–2718.

[153] S. Duan, X. Wang, and H. Zhang, "FIN: Practical signature-free asynchronous common subset in constant time," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2023, pp. 815–829.

[154] L. Yang, S. J. Park, et al., "DispersedLedger: High-throughput Byzantine consensus on variable bandwidth networks," in *Proc. USENIX Symp. Netw. Syst. Des. Implement*., 2022, pp. 493–512.

[155] T. Crain, et al., "Red Belly: A secure, fair, and scalable open blockchain," in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 466–483.

[156] H. Zhang, et al., "How to achieve adaptive security for asynchronous BFT?" *J. Parallel Distrib. Comput*., vol. 169, pp. 252–268, 2022.

[157] S. Duan and H. Zhang, "Foundations of dynamic BFT," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 1317–1334.

[158] C. Liu, et al., EPIC: Efficient asynchronous BFT with adaptive security, in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2020, pp. 437–451.

[159] I. Keidar, et al., "All you need is DAG," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2021, pp. 165–175.

[160] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus," in *Proc. Eur. Conf. Comput. Syst.*, 2022, pp. 34–50.

[161] A. Spiegelman, B. Arun, R. Gelashvili, and Z. Li, "Shoal: Improving DAG-BFT latency and robustness," in *Proc. Int. Conf. Financ. Cryptogr. Data Secur.*, 2024, pp. 92–109.

[162] X. Dai, et al., "LightDAG: A low-latency DAG-based BFT consensus through lightweight broadcast," in *Proc. IEEE Int. Parallel Distrib. Process. Symp*., 2024, pp. 998–1008.

[163] X. Dai, et al., "Wahoo: A DAG-based BFT consensus with low latency and low communication overhead," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp.

7508–7522, 2024.

[164] B. Arun, Z. Li, F. Suri-Payer, S. Das, and A. Spiegelman, "Shoal++: High throughput DAG-BFT can be fast and robust!" in *Proc. USENIX Symp. Netw. Syst. Des. Implement.*, 2025, pp. 813–826.

[165] N. Shrestha, R. Shrothrium, A. Kate, and K. Nayak, "Sailfish: Towards improving the latency of DAG-based BFT," in *Proc. IEEE Symp. Secur. Privacy*, 2025, pp. 1928–1946.

[166] X. Dai, et al., "Remora: A low-latency DAG-based BFT through optimistic paths," *IEEE Trans. Comput.*, vol. 74, no. 1, pp. 57–70, 2025.

[167] D. Malkhi, C. Stathakopoulou, and M. Yin, "BBCA-Chain: Low latency, high throughput BFT consensus on a DAG," in *Proc. Int. Conf. Financ. Cryptogr. Data Secur.*, 2024, pp. 51–73.

[168] D. S. Silva, et al., "Threat adaptive Byzantine fault tolerant state-machine replication," in *Proc. IEEE Int. Symp. Reliable Distrib. Syst*, 2021, pp. 78–87.

[169] M. Castro and B. Liskov, "Proactive recovery in a Byzantine-Fault-Tolerant system," in *Proc. Symp. Oper. Syst. Des. Implement*., 2000.

[170] P. Sousa, A. N. Bessani, et al., "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Trans. Parallel Distrib. Syst*., vol. 21, no. 4, pp. 452–465, 2010.

[171] A. Schiper, "Dynamic group communication," *Distrib. Comput*., vol. 18, pp. 359–374, 2006.

[172] R. Guerraoui, J. Komatovic, et al., "Dynamic Byzantine reliable broadcast [technical report]," *arXiv preprint arXiv:2001.06271*, 2020.

[173] J. Yin, J. P. Martin, et al., "Separating agreement from execution for Byzantine fault tolerant services," in *Proc. ACM Symp. Oper. Syst. Principles*, 2003, pp. 253–267.

[174] Y. Busnel, et al., "On the uniformity of peer sampling based on view shuffling," *J. Parallel Distrib. Comput*., vol. 71, no. 8, pp. 1165–1176, 2011.

[175] T. Bracha, "Asynchronous Byzantine agreement protocols," *Inform. Comput.*,

vol. 75, no. 2, pp. 130–143, 1987.

[176] B. Guo, Y. Lu, et al., "Speeding Dumbo: Pushing asynchronous BFT closer to practice," *IACR Cryptol. ePrint Arch.*, Report 2022/27, 2022.

[177] R. Pass and E. Shi, "The sleepy model of consensus," in *Proc. Adv. Cryptol. ASIACRYPT*, 2017, pp. 380–409.

[178] V. Goyal, et al., "Instant block confirmation in the sleepy model," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2021, pp. 65–83.

[179] D. Malkhi, A. Momose, and L. Ren, "Towards practical sleepy BFT," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2023, pp. 490–503.

[180] X. Ding, J. Guo, et al., "An incentive mechanism for building a secure blockchain-based Internet of Things," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 477–487, 2020.

[181] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–34, 2019.

[182] M. Javaid, A. Haleem, R. P. Singh, R. Suman, and S. Khan, "A review of blockchain technology applications for financial services," *BenchCouncil Trans. Benchmarks Stand. Eval.*, vol. 2, no. 3, p. 100073, 2022.

[183] A. Augusto, et al., "SoK: Security and privacy of blockchain interoperability," *in Proc. IEEE Symp. Secur. Privacy*, pp. 3840–3865, 2024.

[184] A. Lohachab, et al., "Towards interconnected BCs: A comprehensive review of the role of interoperability among disparate blockchains," *ACM Comput. Surveys*, vol. 54, no. 7, pp. 1–39, 2021.

[185] H. Tian, et al., "Enabling cross-chain transactions: A decentralized cryptocurrency exchange protocol," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3928–3941, 2021.

[186] T. Yu, F. Luo, and G. Ranzi, "PCCAE: A protocol for multi-party asset exchange among blockchains," *in Proc. ACM SIGCOMM Workshop Ze-ro Trust Archit. Next Gener. Commun.*, 2024, pp. 13–18.

[187] Y. Xue and M. Herlihy, "Hedging against sore loser attacks in cross-chain

transactions," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2021, pp. 155–164.

[188] M. Herlihy, et al., "Cross-chain deals and adversarial commerce," *The VLDB J.*, vol. 31, no. 6, pp. 1291–1309, 2022.

[189] S. A. Thyagarajan, et al., "Universal atomic swaps: Secure exchange of coins across all blockchains," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 1299–1316.

[190] S. Imoto, et al., "Atomic cross-chain swaps with improved space, time, and local time complexities," *Inf. Comput*., vol. 292, 2023, Art. no. 105039.

[191] Yacov Manevich, et al., "Cross chain atomic swaps in the absence of time via attribute verifiable timed commitments," in *Proc. IEEE 7th Eur. Symp. Secur. Privacy*, 2022, pp. 606–625.

[192] S. You, A. Joshi, A. Kuehlkamp, and J. Nabrzyski, "A multi-party, multi-blockchain atomic swap protocol with universal adaptor secret," *arXiv preprint arXiv:2406.16822.* 2024.

[193] A. Zamyatin, et al., "XCLAIM: Trustless, interoperable, cryptocurrency backed assets," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 193–210.

[194] *Layerzero*. Accessed: Dec. 2025. [Online]. Available: https://layerzero.gitbook.io/

[195] M. Westerkamp, et al., "zkrelay: Facilitating sidechains using zksnark-based chain-relays," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, 2020, pp. 378–386.

[196] L. Breidenbach, et al., "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," *Chainlink Labs*, vol. 1, pp. 1–136, 2021.

[197] P. Sheng, et al., "TrustBoost: Boosting trust among interoperable blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2023, pp. 1571–1584.

[198] M. Westerkamp and A. Küpper, "Instant Function Calls using Synchronized Cross-Blockchain Smart Contracts," *IEEE Trans. Network Service Manag*., vol. 20, no. 3, pp. 2136–2150, 2023.

[199] *Axelar Team*. (2021) Axelar Network: Connecting Applications with blockchain

Ecosystems.

[200] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, “Hermes: Fault-tolerant middleware for blockchain interoperability,” *Future Gener. Comput. Syst*., vol. 129, pp. 236–251, 2022.

[201] P. Zhang, et al., “Cross-chain digital asset system for secure trading and payment,” *IEEE Trans. Comput. Soc. Syst*., vol. 11, no. 2, pp. 1654–1666, 2024.

[202] J. Kwon and E. Buchman. (2019) *Cosmos WhitePaper*. [Online]. Available: https://cosmos.network/whitepaper

[203] A. Xiong, et al., “A notary group-based cross-chain mechanism,” *Digit. Commun. Netw*., vol. 8, no. 6, pp. 1059–1067, 2022.

[204] Z. Liu, et al., “Hyperservice: Interoperability and programmability across heterogeneous blockchains,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., 2019, pp. 549–566.

[205] Y. Liu, et al., “Secure and scalable cross-domain data sharing in zero-trust cloud-edge-end environment based on sharding blockchain,” *IEEE Trans. Dependable Secure Comput*., vol. 21, no. 4, pp. 2603–2618, 2024.

[206] L. Yin, J. Xu, and Q. Tang, “Sidechains with fast cross-chain transfers,” *IEEE Trans. Dependable Secure Comput*., vol. 19, no. 6, pp. 3925–3940, 2022.

[207] P. Gazi, et al., “Proof-of-stake sidechains,” in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 139–156.

[208] L. Yin, et al., “Sidechains with optimally succinct proof,” *IEEE Trans. Dependable Secure Comput*., vol. 21, no. 4, pp. 3375–3389, 2024.

[209] *XCMP*. Accessed: Dec. 2025. [Online]. Available: https://wiki.polkadot.network/

[210] Y. Cao, et al., “MAP the blockchain world: A trustless and scalable blockchain interoperability protocol for cross-chain applications,” *in Proc. ACM Web Conf.*, 2025, pp. 717–726.

[211] T. Xie, et al., “zkBridge: Trustless cross-chain bridges made practical,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 3003–3017.

[212] Y. Chen, A. Asheralieva, and X. Wei, “AtomCI: A new system for the atomic cross-chain smart contract invocation spanning heterogeneous blockchains,” *IEEE Trans. Netw. Sci. Eng*., vol. 11, no. 3, pp. 2782–2796, 2024.

[213] G. Wang, et al., “Exploring blockchains interoperability: A systematic survey,” *ACM Comput. Surveys*, vol. 55, no. 13s, pp. 1–38, 2023.

[214] F. Martinelli and N. Mushegian. (2019) *Balancer: A Non-custodial Portfolio Manager, Liquidity Provider, and Price Sensor.* [Online]. Available: https://balancer. finance/whitepaper

[215] B. Pillai, K. Biswas, Z. Hóu, and V. Muthukkumarasamy, “The burn-to-claim cross-blockchain asset transfer protocol,” in *Proc. Int. Conf. Eng. Complex Comput. Syst.*, 2020, pp. 119–124.

[216] B. Pillai, K. Biswas, Z. Hóu, and V. Muthukkumarasamy, “Burn-to-claim: An asset transfer protocol for blockchain interoperability,” *Comput. Netw*., vol. 200, 2021, Art. no. 108495.

[217] R. Bacho and J. Loss, “On the adaptive security of the threshold BLS signature scheme,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 193–207.

[218] *Ganache*. Accessed: Dec. 2025. [Online]. Available: https://trufflesuite.com/

[219] *Remix*. Accessed: Dec. 2025. [Online]. Available: https://remix.ethereum.org/

[220] S. Zhang and J.-H. Lee, “Analysis of the main consensus protocols of blockchain,” *ICT Express*, vol. 6, no. 2, pp. 93–97, 2020.

[221] X. Wang, S. Duan, J. Clavin, and H. Zhang, “BFT in blockchains: From protocols to use cases,” *ACM Comput. Surv*., vol. 54, no. 10s, Art. no. 209, pp. 1–37, 2022.

[222] J. A. Garay, A. Kiayias, and N. Leonardos, “The bitcoin backbone protocol: Analysis and applications,” *in Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.,* 2015, pp. 281–310.

[223] W. Zou, D. Lo, P. S. Kochhar, et al., “Smart contract development: Challenges and opportunities,” *IEEE Trans. Softw. Eng*., vol. 47, no. 10, pp. 2084–2106,

2021.

[224] B. Xue, S. Deb, and S. Kannan, “BigDipper: A hyperscale BFT system with short term censorship resistance,” *arXiv preprint arXiv:2307.10185*. 2023.

[225] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, “Scalable and probabilistic leaderless BFT consensus through metastability,” *arXiv preprint arXiv:1906.08936*, 2019. [Online]. Available: https://arxiv.org/abs/1906.08936.

[226] A. Wahrstätter, et al., “Blockchain censorship,” in *Proc. ACM Web Conf.*, 2024, pp. 1632–1643.

[227] *Github*. Accessed: Dec. 2025. [Online]. Available: https://GitHub.com/BryantTY/Test-for-RC-system/

[228] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum Project Yellow Paper*, 2014.

[229] J. Camenisch, et al., “Internet computer consensus,” in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2022, pp. 81–91.

[230] E. Androulaki, et al., “Hyperledger fabric: a distributed operating system for permissioned blockchains,” in *Proc. EuroSys Conf.*, 2018, Art. no. 30, pp. 1–15.

[231] T. Yu, et al., “Dual-Blockchain-Based P2P Energy Trading System With an Improved Optimistic Rollup Mechanism,” *IET Smart Grid*, vol. 5, no. 4, pp. 246–259, 2022.

[232] M. Khorasany, Y. Mishra, et al., “A Decentralized Bilateral Energy Trading System for Peer-to-Peer Electricity Markets,” *IEEE Trans. Ind. Electron.*, vol. 67, no. 6, pp. 4646–4657, 2020.

[233] G. Belgioioso, et al., “Operationally-Safe Peer-to-Peer Energy Trading in Distribution Grids: A Game-Theoretic Market-Clearing Mechanism,” *IEEE Trans. Smart Grid*, vol. 13, no. 4, pp. 2897–2907, 2022.

[234] Y. He, et al., “A multi-block ADMM based approach for distribution market clearing with distribution locational marginal price,” *Int. J. Electr. Power Energy Syst.*, vol. 128, 2021, Art. no. 106635.

[235] D. K. Molzahn, et al., “A survey of distributed optimisation and control

algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[236] N. Patari, et al., "Distributed optimisation in distribution systems: Use cases, limitations, and research needs," *IEEE Trans. Power Syst.*, vol. 37, no. 5, pp. 3469–3481, 2022.

[237] J. Liu, et al., "Privacy-Preserving Peer-to-Peer Energy Trading via Hybrid Secure Computations," *IEEE Trans. Smart Grid*, vol. 15, no. 2, pp. 1951–1964, 2024.

[238] J. Yu, Y. Weng, and R. Rajagopal, "PaToPa: A data-driven parameter and topology joint estimation framework in distribution grids," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 4335–4347, 2018.

[239] K. Kuwaranancharoen and S. Sundaram, "On the geometric convergence of byzantine-resilient distributed optimisation algorithms," *SIAM J. Optim.*, vol. 35, no. 1, pp. 210–239, 2025.

[240] Q. Wang, et al., "Review of real-time electricity markets for integrating distributed energy resources and demand response," *Appl. Energy*, vol. 138, pp. 695–706, 2015.

[241] X. Zhou, et al., "Bidirectional Privacy-Preserving Network-Constrained Peer-to-Peer Energy Trading Based on Secure Multiparty Computation and Blockchain," *IEEE Trans. Power Syst*., vol. 39, no. 1, pp. 602–613, 2024.

[242] J. Ping, Z. Yan, and S. Chen, "A privacy-preserving blockchain-based method to optimize energy trading," *IEEE Trans. Smart Grid*, vol. 14, no. 2, pp. 1148–1157, 2023.

[243] C. Mu, et al., "Energy block-based peer-to-peer contract trading with secure multi-party computation in nanogrid," *IEEE Trans. Smart Grid*, vol. 13, no. 6, pp. 4759–4772, 2022.

[244] P. Martins, L. Sousa, and A. Mariano, "A survey on fully homomorphic encryption: An engineering perspective," *ACM Comput. Surv*., vol. 50, no. 6, pp. 1–33, 2017.

[245] S. Chen, et al., “A blockchain consensus mechanism that uses Proof of Solution to optimize energy dispatch and trading,” *Nat. Energy*, vol. 7, no. 6, pp. 495–502, 2022.

[246] F. Mohammadi and M. Saif, “Blockchain technology in modern power systems: A systematic review,” *IEEE Syst., Man, Cybern. Mag.*, vol. 9, no. 1, pp. 37–47, 2023.

[247] M. Zade, et al., “Evaluating the added value of blockchains to local energy markets – Comparing the performance of blockchain-based and centralised implementations,” *IET Smart Grid*, vol. 5, no. 4, pp. 234–245, 2022.

[248] Q. Sun, et al., “Break down the decentralization-security-privacy trilemma in management of distributed energy systems,” *Nat. Commun*., vol. 15, no. 1, 2024, Art. no. 4508.

[249] X. Lu and H. Guo, “Trust-DETM: Distributed energy trading model based on trusted execution environment,” *Mathematics*, vol. 11, no. 13, p. 2934, 2023.

[250] G. Hu, Y. Wu, G. Chen, T. T. A. Dinh, and B. C. Ooi, “Sesemi: Secure serverless model inference on sensitive data,” in *Proc. IEEE 41st Int. Conf. Data Eng. (ICDE)*, 2025, pp. 2201–2214.

[251] S. Jiang, C. Li, H. Li, and R. Lu, “Secure and privacy-preserving energy trading with demand response assistance based on blockchain,” *IEEE Trans. Netw. Sci. Eng*., vol. 11, no. 1, pp. 1238–1250, 2024.

[252] G. Luo, et al., “Efficient load flow for large weakly meshed networks,” *IEEE Trans. Power Syst.*, vol. 5, no. 4, pp. 1309–1316, 1990.

[253] H.-G. Yeh, D. F. Gayme, and S. H. Low, “Adaptive var control for distribution circuits with photovoltaic generators,” *IEEE Trans. Power Syst*., vol. 27, no. 3, pp. 1656–1663, 2012.

[254] B. Houska, J. Frasch, and M. Diehl, “An augmented Lagrangian based algorithm for distributed nonconvex optimisation,” *SIAM J. Optim.*, vol. 26, no. 2, pp. 1101–1127, 2016.

[255] G. Xu, H. Bai, J. Xing, T. Luo, N. N. Xiong, X. Cheng, S. Liu, and X. Zheng,

"SG-PBFT: A secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent Internet of vehicles," *J. Parallel Distrib. Comput.*, vol. 164, pp. 1–11, 2022.

[256] V. Shoup, "Practical threshold signatures," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn*., 2000, pp. 207–220.

[257] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur*., *(CCS)*, 2020, pp. 1575–1590.

[258] H. J. Touma, "Study of the economic dispatch problem on IEEE 30-bus system using whale optimisation algorithm," *Int. J. Eng. Technol. Sci*., vol. 3, no. 1, pp. 11–18, 2016.

[259] "PyCryptodome," GitHub, https://github.com/Legrandin/pycryptodome, accessed Dec. 2025.

[260] R. D. Zimmerman, et al., "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst*., vol. 26, no. 1, pp. 12–19, 2011.

[261] S. Aslam, et al., "A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids," *Renew. Sustain. Energy Rev*., vol. 144, 2021, Art. no. 110992.

[262] Y. He, et al., "Privacy-preserving and hierarchically federated framework for short-term residential load forecasting," *IEEE Trans. Smart Grid*, vol. 14, no. 6, pp. 4409–4423, 2023.

[263] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.

[264] C. Dong, et al., Cheng, and S. Yu, "Privacy-preserving and Byzantine-robust federated learning," *IEEE Trans. Depend. Secure Comput*., vol. 21, no. 2, pp. 889–904, 2024.

[265] X. Guo, et al., "VeriFL: Communication-efficient and fast verifiable

aggregation for federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1736–1751, 2021.

[266] T. Stevens, et al., "Efficient differentially private secure aggregation for federated learning via hardness of learning with errors," in *Proc. 31st USENIX Secur. Symp*., 2022, pp. 1379–1395.

[267] X. Wang, H. Wang, S. Li, and H. Jin, "A reinforcement learning-based online learning strategy for real-time short-term load forecasting," *Energy*, vol. 305, 2024, Art. no. 132344.

[268] Q. Dong, et al., "Short-term electricity-load forecasting by deep learning: A comprehensive survey," *Eng. Appl. Artif. Intell*., vol. 154, 2025, Art. no. 110980.

[269] M. A. T. A. Elgalhud, M. N. Fekri, S. Mir, and K. Grolinger, "Federated online learning for adaptive load forecasting across decentralized nodes," *Int. J. Electr. Power Energy Syst*., vol. 169, 2025, Art. no. 110779.

[270] Q. Mao, et al., "A blockchain-based framework for federated learning with privacy preservation in power load forecasting," *Knowl. Based Syst*., vol. 284, 2024, Art. no. 111338.

[271] N. Jia, et al., "A comprehensive survey on communication-efficient federated learning in mobile edge environments," *IEEE Commun. Surv. Tutorials*, vol. 27, no. 6, pp. 3710–3741, 2025.

[272] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Comput. Surv*., vol. 55, no. 4, pp. 1–35, 2022.

[273] Z. Peng, et al., "VFChain: Enabling verifiable and auditable federated learning via blockchain systems," *IEEE Trans. Netw. Sci. Eng*., vol. 9, no. 1, pp. 173–186, 2022.

[274] Z. Wang, et al., "Incentive mechanism design for joint resource allocation in blockchain-based federated learning," *IEEE Trans. Parallel Distrib. Syst*., vol. 34, no. 5, pp. 1536–1547, 2023.

[275] T. Yu et al. "Blockchain in smart grids: a review of recent developments." in

Emerging Smart Technologies for Critical Infrastructure. Springer, 2023, pp. 23–59.

[276] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 493–506.

[277] W. Dally, "On the model of computation: Point," *Commun. ACM*, vol. 65, no. 9, pp. 30–32, 2022.

[278] A. Reisizadeh, et al., "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Stat.,* 2020, pp. 2021–2031.

[279] R. Dai, L. Shen, F. He, X. Tian, and D. Tao, "DisPFL: Towards communication-efficient personalized federated learning via decentralized sparse training," *arXiv preprint arXiv*:2206.00187, 2022.

[280] M. Kim, et al., "SpaFL: Communication-efficient federated learning with sparse models and low computational overhead," *Adv. Neural Inf. Process. Syst*., vol. 37, pp. 86500–86527, 2024.

[281] Z. Shen, Y. Cai, K. Cheng, P. P. C. Lee, X. Li, Y. Hu, and J. Shu, "A survey of the past, present, and future of erasure coding for storage systems," *ACM Trans. Storage*, vol. 21, no. 1, Art. no. 4, pp. 1–39, 2025.

[282] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Proc. Int. Workshop Public Key Cryptogr.*, 2013, pp. 55–72.

[283] D. He, C. Chen, S. Chan, and J. Bu, "Secure and efficient handover authentication based on bilinear pairing functions," *IEEE Trans. Wireless Commun*., vol. 11, no. 1, pp. 48–53, 2012.

[284] G. Luo, S. Fu, and G. Gong, "Speeding up multi-scalar multiplication over fixed points towards efficient zkSNARKs," *IACR Trans. Cryptogr. Hardw. Embed. Syst*., vol. 2023, no. 2, pp. 358–380, 2023.

[285] T. Yu, F. Luo, and G. Ranzi, "Time efficient, privacy preserving and byzantine tolerant distributed optimisation framework for peer-to-peer energy markets,"

Authorea Preprints, 2025.

[286] Z. Guan, et al., "OPSA: Efficient and verifiable one-pass secure aggregation with TEE for federated learning," *IEEE Trans. Depend. Secure Comput.*, vol. 22, no. 5, pp. 5321–5334, 2025.

[287] F. Mo, et al., "PPFL: Privacy-preserving federated learning with trusted execution environments," in *Proc. 19th Annu. Int. Conf. Mobile Syst., Appl., Serv.*, 2021, pp. 94–108.

[288] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. USENIX Symp. Oper. Syst. Des. Implement.*, vol. 99, 1999, pp. 173–186.

[289] *Klauspost*. Accessed: Dec. 2025. [Online]. Available: https://github.com/klauspost/reedsolomon

[290] *Pymerkle*. Accessed: Dec. 2025. [Online]. Available: https://pypi.org/project/pymerkle/

[291] Kelly, F. P., and R. J. Williams. "Fluid model for a network operating under a fair bandwidth-sharing policy," *Ann. Appl. Probab.*, vol. 14, no. 3, pp. 1055–1083, 2004.

[292] A. Yazdinejad, et al., "A robust privacy-preserving federated learning model against model poisoning attacks," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 6693–6708, 2024.

[293] S. Das and L. Ren, "Adaptively secure BLS threshold signatures from DDH and co-CDH," in *Proc. Adv. Cryptology - CRYPTO 2024, LNCS 14926*, pp. 251–284, 2024.

[294] L. Wang, et al., "Dynamic adaptive cross-chain trading mode for multi-microgrid joint operation," *Sensors*, vol. 20, no. 21, Art. no. 6096, 2020.

[295] N. Junaidi, M. P. Abdullah, B. Alharbi, and M. Shaaban, "Blockchain-based management of demand response in electric energy grids: A systematic review," *Energy Reports*, vol. 9, pp. 5075–5100, 2023.

[296] Y. Wang, H. Xie, X. Sun, L. Tang, and Z. Bie, "A cross-chain enabled day-ahead collaborative power-carbon-TGC market," *Energy*, vol. 258, Art. no. 124881,

2022.

[297] Z. Jiang, Y. Guo, and J. Wang, “Dynamic operating envelopes embedded peer-to-peer-to-grid energy trading,” *Appl. Energy*, vol. 377, Part B, 2025, Art. no. 124554.

[298] B. McMahan, et al., “Communication-efficient learning of deep networks from decentralized data,” in *Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2017, pp. 1273–1282.