Towards Blockchain Systems for Resource-Constrained Devices: Scalability and Data Redaction

A thesis submitted by

Imane El Abid

In partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in Computer Science

at Mohammed VI Polytechnic University,

College of Computing,

Benguerir, Morocco.

Presented before the following Board of Examiners:

Prof. Youssef Iraqi, President of the Jury & Examiner.
Prof. Hugues Fauconnier, Referee.
Prof. Achour Mostefaoui, Referee.
Prof. Maria Potop-Butucaru, Referee.
Prof. Ismail Berrada, Examiner.
Prof. Mustapha Hedabou, Supervisor
Prof. Yahya Benkaouz, Co-supervisor.

UM6P, BenGuerir, 26 June, 2026

Acknowledgements

In the name of Allah, the Most Gracious, the Most Merciful.

First and foremost, I thank Allah Almighty, who blessed me with strength, patience, and knowledge to complete this work. His guidance and mercy made this journey possible.

This thesis is dedicated to the cherished memory of my late professor, *Noureddine Idboufker*. May the ajr of his teachings continue to bless him eternally.

I am grateful to all those who, in one way or another, helped me reach this milestone. This achievement is not mine alone.

I thank the members of my thesis committee, for taking the time to evaluate my work and for their constructive feedback.

My deepest thanks go to my family.
To my dear parents, your patience, sacrifices, and constant *duaa* were my source of strength. Your prayers were my compass through the hardest moments.
To my beloved brother and sister, thank you for being my constant source of laughter and distraction. You reminded me that even in the middle of research chaos, there is always time to laugh.
To my distant friends, *Imane*, *Meryama*, and *Nihad*, thank you for your "survival check-in" calls, your listening ears, and your healing words. Your presence reached me across the miles.
To my best friend, *Kaoutar*, thank you for being my unofficial therapist, for your honest words, and for lifting me up when I felt overwhelmed.

Special thanks to my classmates and cohort members, especially *Meryem*, *Yunusa*, and *Karim*, for the discussions, shared laughter, and moments of joy that made this path less heavy.

Most especially, I owe a profound debt of gratitude to my colleague *Boubouh Karim*. Your collaborations, technical expertise, and late-night feedback sessions shaped this work in ways words cannot fully capture. Beyond the research, your constant support and encouragement carried me through. This thesis is, in many ways, the fruit of our partnership.

Finally, I take full responsibility for any errors or shortcomings in this work, while acknowledging that its strengths and any merit it may possess come first from the grace of Allah, and then from the support of the people He blessed me to have beside me.

Al-Hamdulillahi Rabbil-Alameen.

BenGueir, 26 June 2026

Abstract

The increasing reliance on blockchain technology has highlighted its promise as a decentralized and trustless model for digital interactions. However, mainstream blockchain frameworks remain inaccessible to resource-constrained devices such as smartphones and IoT devices, due to high computational costs, rigid immutability, and heavy storage demands. This disconnect undermines the vision of inclusive decentralization and limits blockchain's scalability and applicability in heterogeneous environments. This thesis addresses these challenges by investigating the integration of blockchain with resource-constrained devices in a scalable and adaptable manner that supports controlled data modification. Specifically, three research directions are pursued: (i) overcoming energy and scalability barriers in consensus protocols, (ii) enabling efficient and autonomous participation of low-resource devices without reliance on centralized or specialized full nodes, and (iii) reconciling blockchain immutability with the need for adaptable data management that allows secure and controlled modification. To this end, the thesis introduces three key contributions. First, LighTx, a lightweight transaction execution framework that minimizes computational complexity by combining a lightweight reliable-broadcast-based Proof-of-Bandwidth consensus with a decentralized reputation system that ensures fairness and Sybil resistance. Second, PocketChain, a novel blockchain architecture specifically designed to enable the active participation of resource-constrained nodes in consensus and validation. PocketChain employs a lightweight consensus mechanism that minimizes computational and communication overhead, allowing devices like smartphones to contribute to network security without excessive battery or bandwidth consumption. Third, we confront the immutability paradox by exploring controlled data redaction mechanisms that enable necessary data modifications, while preserving the ledger's critical security properties. Through these contributions, the thesis advances the design of scalable, adaptable blockchain systems capable of operating across heterogeneous networks that include resource-constrained devices as active participants, thereby extending their deployment into emerging domains such as edge computing, digital identity management, and decentralized IoT ecosystems.

Keywords: Distributed Ledger technology, Scalability, Resource-constrained devices, Lightweight consensus, Redactable blockchain.

Résumé

Le recours croissant à la technologie blockchain a mis en évidence son potentiel en tant que modèle décentralisé et sans tiers de confiance pour les interactions numériques. Cependant, les modèles blockchain traditionnels restent inaccessibles aux appareils aux ressources limitées, tels que les smartphones et les appareils IoT, en raison de coûts de calcul élevés, d'une immuabilité rigide et d'exigences de stockage importantes. Ce décalage contredit la vision d'une décentralisation inclusive et limite l'évolutivité et l'applicabilité de la blockchain dans des environnements hétérogènes. Cette thèse aborde ces défis en étudiant l'intégration de la blockchain à des appareils aux ressources limitées, de manière évolutive et adaptable qui prend en charge la modification contrôlée des données. Plus précisément, trois axes de recherche sont explorés : (i) surmonter les obstacles liés à l'énergie et à l'évolutivité dans les protocoles de consensus, (ii) permettre une participation efficace et autonome des appareils à faibles ressources sans dépendre de nœuds complets centralisés ou spécialisés, et (iii) concilier l'immuabilité de la blockchain avec la nécessité d'une gestion adaptable des données qui permette une modification sécurisée et contrôlée. À cette fin, la thèse présente trois contributions clés. Premièrement, LighTx, un cadre d'exécution de transactions léger qui minimise la complexité computationnelle en combinant un consensus Proof-of-Bandwidth fiable et léger, basé sur la diffusion, avec un système de réputation décentralisé qui garantit l'équité et la résistance à Sybil. Deuxièmement, PocketChain, une nouvelle architecture de blockchain spécialement conçue pour permettre la participation active de nœuds aux ressources limitées au consensus et à la validation. PocketChain utilise un mécanisme de consensus léger qui minimise la surcharge computationnelle et de communication, permettant à des appareils tels que les smartphones de contribuer à la sécurité du réseau sans consommation excessive de batterie ou de bande passante. Troisièmement, nous affrontons le paradoxe de l'immuabilité en explorant des mécanismes de rédaction contrôlée des données qui permettent les modifications nécessaires, tout en préservant les propriétés de sécurité essentielles du registre. Grâce à ces contributions, la thèse fait progresser la conception de systèmes blockchain évolutifs et adaptables, capables de fonctionner sur des réseaux hétérogènes qui incluent des appareils aux ressources limitées en tant que participants actifs, étendant ainsi leur déploiement à des domaines émergents tels que l'informatique de pointe, la gestion d'identité numérique et les écosystèmes IoT décentralisés.

Mots-clés : Technologie de registre distribué, Scalabilité, Appareils à ressources limitées, Consensus léger, Blockchain modifiable.

الخلاصة

أدى الاعتماد المتزايد على تقنية قواعد البيانات المتسلسلة (’سلسلة الكتل‘) إلى إبراز إمكاناتها كنموذج لامركزي ولا يحتاج إلى ثقة للتعاملات الرقمية. ومع ذلك، لا تزال أطر عمل قواعد البيانات المتسلسلة السائدة غير متاحة للأجهزة ذات الموارد المحدودة مثل الهواتف الذكية ونُظم إنترنت الأشياء، وذلك بسبب التكاليف الحسابية المرتفعة، وعدم قابليتها للتغيير، ومتطلبات التخزين الكبيرة. ويقوض هذا الانفصال رؤية اللامركزية الشاملة ويحد من قابلية قواعد البيانات المتسلسلة للتطوير والتطبيق في بيئات غير متجانسة.

تتناول هذه الأطروحة هذه التحديات من خلال دراسة إمكانية دمج تقنية قواعد البيانات المتسلسلة مع الأجهزة ذات الموارد المحدودة بطريقة قابلة للتوسع والتكيف تدعم تعديل البيانات بشكل متحكم فيه. وعلى وجه التحديد، يتم اتباع ثلاثة اتجاهات بحثية: (1) التغلب على حواجز الطاقة وقابلية التوسع في بروتوكولات التوافق؛ (2) تمكين المشاركة الفعالة والمستقلة للأجهزة ذات الموارد المحدودة دون الاعتماد على العقد المركزية أو المتخصصة الشاملة؛ (3) التوفيق بين عدم قابلية التغيير قواعد البيانات المتسلسلة والحاجة إلى إدارة بيانات قابلة للتكيف تسمح بالتعديل الآمن والمحكوم. ولتحقيق هذه الغاية، تقدم الأطروحة ثلاث مساهمات رئيسية. أولاً، LighTx، وهو إطار عمل خفيف الوزن لتنفيذ المعاملات يقلل من التعقيد الحسابي عن طريق الجمع بين نظام توافق Proof-of-Bandwidth الخفيف الوزن والموثوق القائم على البث الموثوق به ونظام سمعة لامركزي يضمن العدالة ومقاومة Sybil. ثانيًا، PocketChain، وهي منظومة جديدة لقواعد البيانات الموزعة مصممة خصيصًا لتمكين المشاركة الفعالة للأجهزة ذات الموارد المحدودة في عملية التوافق والتحقق. تستخدم niahCtekcoP آلية توافق خفيفة الوزن تقلل من عبء الحساب والاتصال، مما يسمح لأجهزة مثل الهواتف الذكية بالمساهمة في أمن الشبكة دون استهلاك مفرط للبطارية أو النطاق الترددي. ثالثًا، نواجه مفارقة عدم قابلية التغيير للبيانات المسجلة على السلسلة من خلال استكشاف آليات تحرير البيانات الخاضعة للرقابة التي تسمح بإجراء التعديلات الضرورية على البيانات، مع الحفاظ على الخصائص الأمنية الهامة لسجل البيانات.

من خلال هذه المقترحات، تقدم الأطروحة تصميم أنظمة قواعد البيانات المتسلسلة القابلة للتطوير والتكيف وقادرة على العمل عبر شبكات غير متجانسة تضم أجهزة محدودة الموارد كمشاركين نشطين، وبالتالي توسيع نطاق نشرها في مجالات ناشئة مثل الحوسبة الطرفية وإدارة الهوية الرقمية وأنظمة إنترنت الأشياء اللامركزية.

كلمات مفتاحية: قواعد البيانات الموزعة، قابلية التوسع، الأجهزة ذات الموارد المحدودة، التوافق الخفيف، قواعد البيانات القابلة للتعديل.

Contents

List of Figures

List of Tables

LIST OF TABLES

List of Abbreviations

ABET Attribute-Based Encryption with black-box Traceability

ABTT Attribute-Based Traitor Tracing

ASCS Accountable and Sanitizable Chameleon Signatures

ASICs Application-Specific Integrated Circuits

ATM Automated Teller Machine

AWS Amazon Web Services

BFT Byzantine Fault Tolerance

BRB Byzantine Reliable Broadcast

CAP Consistency-Availability-Partition Tolerance

CH Chameleon Hash

CHET Chameleon Hashes with Ephemeral Trapdoors

CP-ABE Ciphertext-Policy Attribute-Based Encryption

DAG Directed Acyclic Graph

DAO Decentralized Autonomous Organization

Dapps Decentralized Applications

DCP-ABE Decentralized Ciphertext-Policy ABE

DKG Distributed Key Generation

DPSS Dynamic Proactive Secret Sharing

DS Digital Signature

EVM Ethereum Virtual Machine

FLP Fischer-Lynch-Paterson

GCP Google Cloud Platform

GDPR General Data Protection Regulation

GPU Graphics Processing Units

LIST OF ABBREVIATIONS

IBS Identity-Based Signatures

IoT Internet of Things

IPFS InterPlanatery File System

LRRS Linkable and Redactable Ring Signature

LSS Linear Secret Sharing

MOMs Memory Optimization Modes

MPC Multi-Party Computation

NITCH Non-Interactive Threshold Chameleon Hash

NIZK Non-interactive zero-knowledge proofs

NLSS Non-Linear Secret Sharing

P2P Peer-to-Peer

PBFT Practical Byzantine Fault Tolerance

PKE Public Key Encryption

PoB Proof-of-Bandwidth

PoS Proof-of-Stake

PoW Proof-of-Work

PTS Privileged Token Service

RCHET Revocable Chameleon Hashes with Ephemeral Trapdoors

RSA Rivest-Shamir-Adleman

RTT Round Trip Time

sBRB Scalable Byzantine Reliable Broadcast

SMR State-Machine Replication

SPOF Single Point of Failure

SPV Simplified Payment Verification

SSI Self-Sovereign Identity

TPS Transactions per Second

TUCH Time Updatable Chameleon Hash

URLLC Ultra-Reliable Low-Latency Communication

UTXO Unspent Transaction Output

VBChs Virtual Broadcast Channels

LIST OF ABBREVIATIONS

VDF Verifiable delay function

VRF Verifiable Random Functions

zk-SNARK Zero-Knowledge Succinct Non-Interactive Argument of Knowledge

ZKP Zero Knowledge Proofs

1 Introduction

Human trading activity began with barter, a decentralized system of direct peer-to-peer exchange fundamentally limited by the ”double coincidence of wants”. Societies progressed through commodity money and precious metals, eventually transitioning to paper and fiat currency. This evolution led to the rise of modern centralized banking, which, while improving efficiency and stability, placed monetary authority within governments and trusted institutions. However, this power concentration introduced several vulnerabilities, including high-value breach targets, single points of failure, and high latency due to hierarchical intermediaries. The fragility of this system was exposed by the 2008 global financial crisis, triggering a severe loss of public trust [142].

In response, Bitcoin emerged in 2008 as a decentralized digital currency [143], introducing a model of algorithmic trust where anyone could join the network, verify transaction authenticity, and rely on a tamper-resistant ledger secured by consensus. Blockchain, the underlying data structure, enabled distributed transaction validation without intermediaries, shifting toward decentralized control. Yet, current blockchain implementations are constrained by their own demanding requirements, substantial computational power, constant connectivity, and considerable storage capacity. These demands translate into high energy consumption and infrastructure costs, creating a high barrier to entry that effectively excludes resource-constrained devices (like mobile phones and IoT sensors) and inadvertently reinforces a tendency toward centralization. As blockchain technology extends to the network edge, its traditional assumptions regarding resource efficiency and scalability no longer hold.

This thesis challenges the traditional blockchain paradigm built on high-capacity nodes and energy-intensive operations by redefining how such systems can operate in heterogeneous, resource-constrained environments. It addresses foundational limitations in consensus efficiency, decentralization, and computational or storage overhead through lightweight consensus mechanisms that preserve decentralization without costly computation, adaptive participation models that scale with device capability, and redaction techniques that enable verifiable data correction while maintaining ledger integrity. Ultimately, this research enables all devices, regardless of capacity, to participate actively in the creation and preservation of shared state across the network.

1.1 The evolution of the Blockchain landscape

The conceptual foundations of blockchain technology emerged decades before its formal inception. Core cryptographic primitives, including public-key encryption [54], hash functions [160], and Merkle trees [138], have established the technical prerequisites for secure and tamper-evident distributed ledgers.

In 1992, Dwork and Naor introduced the concept of *pricing via processing* mechanism [61]. Their scheme proposed that requiring participants to perform a quantifiable yet nontrivial amount of computation before sending a message could make malicious or excessive use economically infeasible. The concept of computational pricing laid the conceptual base for what would later be termed Proof-of-Work (PoW) [100]. The idea was further developed in the Hashcash proposal formalized in 2002 [12], as a practical method to prevent system abuse through computational costs. Users were required to solve a moderately hard computational puzzle to access a service. Both solutions were originally designed to combat email spam by making mass messaging prohibitively expensive.

Around the same time, in the late 1990s, Szabo introduced Bit Gold, a theoretical protocol for a decentralized digital currency [174]. In this design, participants earned currency by solving cryptographic puzzles, with each currency unit having a value proportional to the computational effort expended to solve it. Bit Gold proposed theoretical solutions to several key challenges such as creating unforgeable digital assets, establishing verifiable transaction records, and restricting the issuance of new currency without centralized control. Although never implemented, Bit Gold anticipated many core principles later realized in blockchain systems, particularly the use of computational puzzles for currency issuance and the reliance on a public transaction ledger.

These components, initially developed for different purposes, converged in 2008 with the publication of the Bitcoin whitepaper by an anonymous group called Satoshi Nakamoto [143]. The proposal integrated earlier cryptographic concepts into the first practical application of blockchain technology. Bitcoin architecture addressed the main barrier to decentralized digital payments, which is the double-spending. Unlike physical cash, digital assets can be copied. Without a trusted central authority to verify transactions, the same digital currency could be spent multiple times. The implementation of Bitcoin established a public ledger for recording bitcoin transactions without requiring trust in central authorities including banks and governments for validation.

At its core, blockchain offers an immutable ledger secured by robust cryptography, a decentralized architecture, powered by consensus mechanisms like Proof-of-Work (PoW) to validate transactions and protect the network against attacks. Additionally, Bitcoin solved the double-spending problem through a distributed timestamp server that created an immutable record of transaction chronology. The system also incorporated economic incentives through mining rewards, aligning network security with participant self-interest. The launch of bitcoin in 2009 demonstrated that decentralized digital currencies could work in practice, not just in theory.

Over the next few years, other cryptocurrencies emerged, including Litecoin [126], Ripple [79], and Dogecoin [55]. These cryptocurrencies aimed to improve the blockchain technology by offering faster transactions, reduced energy consumption, or more flexible consensus mechanisms. The subsequent evolutionary leap came with the introduction of Ethereum [190]. Ethereum incorporated the concept of smart contracts, which are self-executing agreements that automatically execute when predefined conditions are met [173]. Smart contracts transformed the blockchain from a specialized transaction ledger into a general-purpose computational platform.

Early blockchains like Bitcoin and Ethereum relied on Proof of Work consensus, which provided security but suffered from computational intensity and low throughput [143, 190, 188]. As blockchain transitions from its early experimental phase toward greater maturity, it faces critical questions about scalability, energy consumption, governance, and regulatory integration. To alleviate these limitations, various optimization techniques have evolved to address scalability, efficiency, and energy consumption challenges.

Off-chain solutions such as state channels [151] and sidechains [13] emerged as a solution that handle certain transactions away from the main blockchain to reduce congestion and improve speed and cost-efficiency. More recently, Ethereum 2.0 introduced a shift to a proof-of-stake consensus model, offering a faster and more energy-efficient alternative [70, 33]. In parallel, advanced scaling strategies like sharding have been proposed and implemented, which divide the blockchain into smaller segments (shards) that process transactions simultaneously [197, 109].

The evolution of blockchain reflects an ongoing pursuit to balance decentralization, security, and efficiency. Each generation has extended the technology's application, from secure digital cash systems to programmable platforms enabling trustless computation. Nevertheless, fundamental trade-offs persist. The central challenge now is no longer proving feasibility but achieving scalable, sustainable performance. The future of blockchain depends on whether emerging architectures can deliver on the potential of blockchain while addressing computational overhead, environmental impact, and economic viability at scale.

1.2 Problem Statement

The digital ecosystem is expanding toward decentralized infrastructures. Billions of interconnected devices, from powerful cloud servers to lightweight edge and IoT nodes, now participate in data exchange and distributed computation [182]. Blockchain technology seems capable of supporting this model given its foundation in distributed consensus and trustless participation. However, in practice, several incompatibilities become evident, due to extensive resource demand of blockchain protocols. The core blockchain principles, like cryptographic verifications, redundant state replication, and Byzantine fault tolerance consensus mechanisms impose considerable overheads [121]. These requirements assume that all participating nodes possess sufficient processing power, persistent storage, stable network connectivity, and reliable energy supply [76].

Existing blockchain systems derive their security and consistency guarantees from implicit homogeneity assumptions. When deployed across hybrid topologies comprising loosely connected, resource-diverse participants, these systems degrade predictably. As a result, powerful nodes (e.g., servers, high-end desktop computers) operate as full nodes maintaining complete ledger copies and actively participating in consensus, while light nodes (e.g., smartphones, mobile wallets) store only essential data with limited validation capabilities. Resource-constrained devices (e.g., sensors, IoT devices), unable to meet the computational and storage demands, are relegated to passive clients. The resulting hierarchy undermines network decentralization and fairness, allowing computational power to determine control.

This thesis investigates the central research problem of *scaling blockchain systems to enable resource-constrained devices to act as active participant by overcoming inherent limitations in consensus protocols, data immutability, and resource barriers*.

We decompose this multifaceted research problem into three research questions:

RQ1: ***Energy and Scalability Barriers.*** Blockchain systems face critical scalability and sustainability

challenges due to energy-intensive consensus mechanisms like PoW. In particular, Bitcoin's PoW consensus incentivizes specialized ASIC hardware and concentrates mining power in large-scale mining pools that ordinary devices cannot compete with [46]. Furthermore, the energy footprint of the PoW is staggering; Bitcoin mining alone consumes over 100 TWh annually, comparable to the electricity usage of entire nations, raising severe environmental concerns [21]. From a performance perspective, the performance bottlenecks inherent to current blockchain implementations limit their wider applicability. For instance, Bitcoin sustains an average throughput of only 7 TPS [20], far below rate required for large-scale payment or data-sharing systems. Such low throughput, weak scalability and reliance on resource-intensive consensus, limits efficient data processing and broader network expansion. These issues stem from inherent trade-offs in the security-decentralization-scalability trilemma, where protocols like PoW prioritize security at the cost of energy waste, while alternatives like Proof-of-Stake (PoS) risk centralization through wealth concentration. So, *how can blockchain consensus mechanisms be redesigned to achieve scalability without sacrificing security or decentralization?*

RQ2: ***Node Resource Constraints.*** Current blockchain implementations exhibit a fundamental contradiction between theoretical decentralization objectives and practical network architecture. System decentralization cannot be achieved by a subset of powerful nodes running resource-intensive consensus protocols; decentralization requires active participation of the majority of nodes. The challenge lies in the fact that these nodes may include resource-constrained devices such as mobile phones, and edge devices that possess limited computational capacity, storage, and network bandwidth. Blockchain networks typically enforce a binary node hierarchy with sharply different capabilities and trust assumptions. Full nodes require high storage, bandwidth, and computational resources, and light nodes limited to partial validation that must rely on trusted full nodes. The absence of mobile-friendly protocols not only creates an accessibility gap but also undermines the core principle of distributed trust, potentially concentrating power in the hands of well-resourced nodes. So, *How can consensus mechanisms be adapted to accommodate efficient participation of low-resource devices without compromising network security or decentralization objectives?*

RQ3: ***Rigid Immutability.*** Blockchain technology was initially introduced for money transactions, where immutability serves as a cornerstone feature ensuring financial integrity and tamper-proof records. However, blockchain has since gained wide adoption across many fields beyond cryptocurrency, including supply chain, healthcare, identity management, and data provenance systems. Occasionally, these applications need to correct errors, remove outdated information, or comply with privacy regulations. The tension is particularly relevant at the network edge, where resource-constrained devices produce continuous data streams that may contain errors, or privacy-sensitive content. Moreover, these devices lack storage capacity for expanding immutable ledgers, which further restricts their long-term participation. The fundamental tension emerges from the conflicting nature of immutability. Immutability provides security by making unauthorized tampering prohibitively costly. The same property creates inflexibility by preventing legitimate modifications even when they are necessary and authorized. Applications requiring data updates, corrections, or privacy adjustments encounter significant challenges in adopting blockchain systems due to their rigid immutability. Henceforth, the immutability barrier hinders the adoption of blockchain in scenarios demanding such flexibility. So, *how can blockchain immutability be reconciled with the need for controlled data redaction?*

1.3 Thesis Objectives

The main objective of this thesis is to conceive a novel framework for blockchain systems that resolves the scalability, security, and decentralization challenges inherent in the blockchain trilemma. The primary focus of this work is the consensus protocol, which constitutes the defining component of blockchain systems and their principal bottleneck. Consensus mechanisms determine how distributed nodes agree on the state of the ledger without central authority. They directly impact system performance, security guarantees, and the practical balance between decentralization and scalability.

Our motivation is to harness the collective computational power of the majority of nodes in the network, instead of relying on a privileged subset of high-resource operators. According to this vision, the blockchain does not depend on isolated clusters of high-performance validators, but aggregates the capacity of thousands or even millions of heterogeneous devices. Devices that often remain idle for most of their operational lifespan, such as smartphones, IoT sensors, personal laptops, and commodity desktops, become active participants in maintaining security and consistency guarantees. The objective is to establish a fair and inclusive blockchain ecosystem where all participants can contribute and benefit from distributed ledger technology, as illustrated graphically in Fig. 1.1.

We enable resource-constrained devices to engage in transaction validation and block formation without prohibitive energy or processing costs. The system prioritizes fairness through participation commitment instead of computational resources, eliminating the barriers that exclude most devices from current blockchain implementations. Simultaneously, we focus on establishing flexible mechanisms that allow the system to handle personal data responsibly without compromising the integrity of the ledger, making it suitable for a wider range of societal applications.

In a nutshell, we aim to leverage the computational power of user-owned devices to make blockchain more accessible, efficient, and closer to its point of use. The result is a robust infrastructure that is lightweight and efficient enough for pervasive adoption and high-assurance applications. Through rigorous theoretical analysis and implementation, we address the feasibility of designing a lightweight blockchain protocol that preserves scalability and flexibility across heterogeneous networks of resource-constrained devices.

1.4 Contributions

This dissertation advances the state-of-the-art of blockchain technology through three main contributions. These contributions form a comprehensive approach to creating an inclusive blockchain ecosystems that function effectively across heterogeneous computing environments:

- We propose LighTx, a lightweight and scalable transaction transfer system that fundamentally addresses the resource-intensive nature of traditional consensus protocols. The system decouples consensus from computational power by introducing a resource-efficient Proof-of-Bandwidth–based reputation mechanism that ensures fairness and Sybil resistance. LighTx demonstrates significant potential for enabling efficient distributed ledger operations on devices with limited processing power, effectively lowering obstacles to broader blockchain participation.
- We address the central problem of this thesis by highlighting the limitations of existing mobile blockchain solutions. We present PocketChain a resource-efficient blockchain that enables active participation of resource-constrained nodes (especially mobile phones) in transaction validation, block proposal, and consensus activities without excessive battery drain or bandwidth consumption.

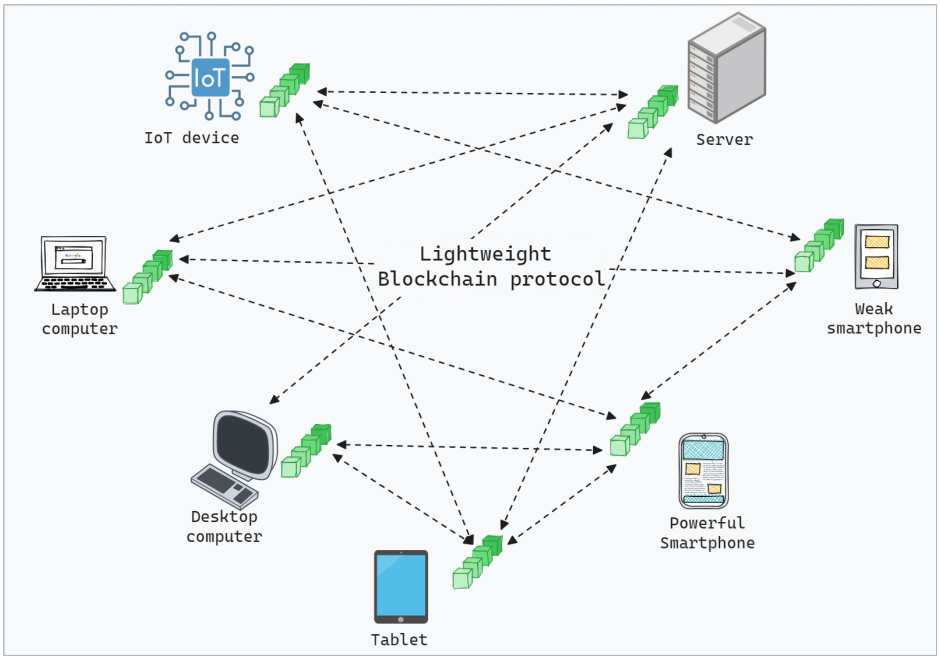


Figure 1.1: A heterogeneous autonomous blockchain system consisting of different node with different hardware capacities

The proposed architecture acknowledges the inherent constraints of mobile environments and employs a lightweight consensus mechanism that minimizes message exchange and computational overhead. Most importantly, PocketChain promotes decentralization and fairness and prevents bottlenecks and enhances network resilience, regardless of individual device capabilities.

- We tackle the blockchain immutability paradox, which simultaneously represents both a core strength and a practical limitation in many application domains. A systematic exploration of various redaction methodologies yields mechanisms for controlled data modification while preserving essential security properties. These findings provide insights for integrating blockchain into applications that require dynamic data management.

1.5 Publications

The main results presented in this thesis are based on the following original publications.

- El Abid Imane, and Yahya Benkaouz. "**Lightx: a lightweight transactions transfer system.**" 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2021. (*Chapter 3*)
- El Abid Imane, Yahya Benkaouz, and Ahmed Khoumsi. "**Lightx: A lightweight proof-of-bandwidth transactions transfer system.**" International Conference on Networked Systems. Springer International Publishing, 2021. (*Chapter 3*)
- El Abid Imane, Karim Boubouh, Yahya Benkaouz. "**PocketChain: Redefining Blockchain Integration with Resource-Constrained Devices**", Future Generation Computer Systems, 2025.

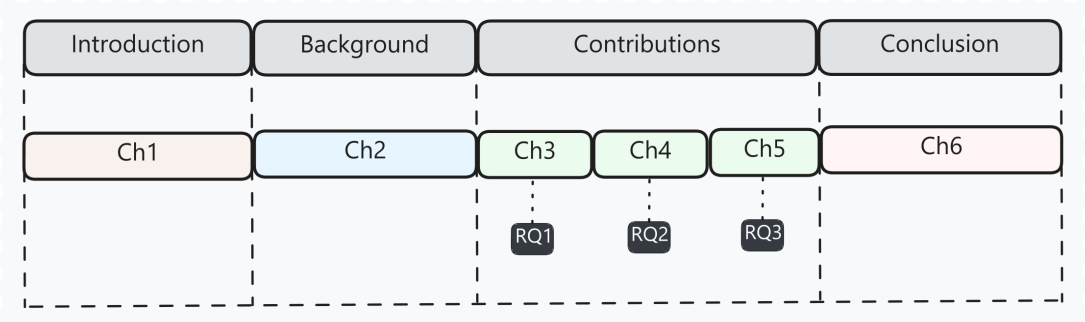


Figure 1.2: Thesis structure

(*Chapter 4*)

- El Abid Imane, and Yahya Benkaouz. "**A Comparative Analysis of Blockchain Redaction Techniques.**" 2024 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS). IEEE, 2024. (*Chapter 5*)
- El Abid Imane, Karim Boubouh, and Yahya Benkaouz. "**Beyond Immutability: A Comprehensive Review of Redactable Blockchain Systems**". Under revision. (*Chapter 5*)

1.6 Roadmap

This thesis is structured into six comprehensive chapters, each methodically synthesizes our contributions toward specific challenges in creating and scaling lightweight blockchain systems that support resource-constrained environments. The progression of these chapters, illustrated in Fig. 1.2, begins by establishing the theoretical foundations, presents our core contributions, and concludes with a synthesis of findings. The detailed structure of each chapter is outlined below.

- Chapter 1 introduces the research context and the main question of this research study.
- Chapter 2 provides the necessary background, detailing the essentials of blockchain technology, its key characteristics, and the challenges of implementing it on resource-constrained devices.
- Chapter 3 describes the design and functionalities of LightTx, a novel, lightweight, and flexible solution that facilitates efficient and secure blockchain operations on resource-constrained nodes.
- Chapter 4 returns to practical considerations by addressing the challenges involved in designing and implementing blockchain architectures specifically for resource-constrained devices.
- Chapter 5 investigates the pioneering redaction methodologies for dynamic data management, thoroughly analyzing the compromises inherent in introducing mutability to blockchain systems.
- The thesis concludes with Chapter 6, summarizing our key findings and their broader implications. This final chapter outlines potential directions for future research and emphasizes the importance of our work in advancing the application of decentralized technologies in the coming years.

2 Background

Modern society is built on centralized systems; governments regulate identity and law, banks govern financial transactions, and corporations like Google and Meta control vast payloads of data and communication. These systems rely on trusted intermediaries to enforce rules and manage resources on behalf of users. Although such centralization enables unified control and simplicity, its exposes inherent vulnerabilities: high-profile data breaches, algorithmic bias, censorship of dissent, and systemic failures like the 2008 financial crisis [1] reveal a fragile foundation. Centralized architectures concentrate power, creating a single point of failure that jeopardize privacy, security, and autonomy. To mitigate these risks, distributed systems emerged, dispersing computation and data across multiple nodes. Technologies like cloud computing and distributed databases improved scalability and fault tolerance, for example, Amazon Web Services (AWS) replicates data globally to ensure uptime. However, distribution alone does not guarantee decentralization. Many distributed systems remain under centralized control. AWS servers are still governed by Amazon, and users must trust the operator to act transparently. This ”pseudo-decentralization” fails to address core issues of power imbalance and opaque governance.

Decentralization reduces or removes unilateral control by a single authority; in fully decentralized systems, including Peer-to-Peer (P2P) networks and blockchain-based protocols, participants collectively enforces the system rules. Here, no single entity controls data flow or decision-making; instead, trust is coordinated through cryptographic verification and aligned by economic incentives. Decentralization promises censorship resistance, user sovereignty, and resilience against collusion, yet achieving these properties requires reconsidering how systems coordinate trust at scale.

This chapter analyzes the limitations of centralization and pseudo-distribution, highlights decentralization as a necessary evolution promoting blockchain for building transparent, and resilient infrastructures. The chapter further introduces the fundamental theoretical concepts, consensus mechanisms, and cryptographic building blocks that underpin decentralized systems, establishing the background required to comprehend the contributions of this thesis.

[1] https://www.rba.gov.au/education/resources/explainers/the-global-financial-crisis.html

2.1 Centralization Dilemma in Modern Digital Ecosystems

The evolution of digital infrastructure has been predominantly shaped by centralization, due to their efficiency, ease of deployment, and historical precedence. Centralized architectures provide an abstracted complexity, enable straightforward regulatory compliance, and reduce coordination overhead, making them the default choice for early systems.

The banking sector provides a particularly illustrative example of these architectural patterns. As shown in Fig. 2.1, traditional banking systems rely on a central bank to govern and regulate financial flows. This structure demonstrates a strict, top-down flow of authority. The central bank sits at the top, setting monetary policy and regulating the commercial banks below. Commercial banks, in turn, act as intermediaries for the users at the base of the hierarchy. Banking systems employ centralized ledgers for transaction processing, account management, and regulatory compliance. In such systems, financial institutions act as trusted intermediaries, ensuring the integrity and security of transactions through a hierarchical authorization model. This centralized architecture enforces sequential transaction processing, where operations are validated in order by a central entity, ensuring strong consistency but also limiting scalability and resilience. Although banks have modernized their services by offering distributed access points such as ATMs, online portals, and mobile apps, these interfaces merely duplicate the entry points to the same central infrastructure. The ledger itself remains centralized, and client interactions are restricted to roles defined by central authorities.

Even with the rise of the fintech revolution, centralization persists as the dominant model. Modern digital platforms like PayPal, Revolut, and Alipay provide near-instant digital payments and seamless user interfaces, fundamentally reshaping how individuals interact with money [71, 123]. These platforms digitize financial activity but all their transactions are still routed through and verified by their private servers on closed, proprietary networks. In this sense, what fintech systems have achieved is distribution of access, not decentralization of control.

The trajectory toward centralization continued into the early 21st century, with the rise of large technology corporations such as Amazon, Google and Meta. Big tech platforms demonstrate the same pattern observed in financial systems. For decades, computational resources and control steadily migrated away from end-users and concentrated in massive cloud data centers operated by a handful of major corporations [51]. Although these platforms operate on globally distributed infrastructures, the underlying control remains centralized, to aggregate unprecedented amounts of user data, process transactions on a global scale, and monitor activity with reduced latency.

Centralized systems offer operational advantages through unified control. Updates and security policies deploy instantly across the entire infrastructure, while concentrated computational resources minimize latency by routing requests to powerful servers. Nonetheless, centralization simultaneously creates a single point of failure, introduces severe privacy and security vulnerabilities, and concentrates power over data in the hands of a few entities. These issues manifest in several ways:

- Centralized architectures are attractive targets for malicious actors, since a successful attack on central components can trigger cascading failures across the entire system. Even in modern implementations that geographically distribute workloads, centralized control over global configuration management, deployment, and authentication services reintroduces fundamental Single Point of Failure (SPOF) [176].
- Centralized systems often lack transparency, as they are closed-source, with control limited to a

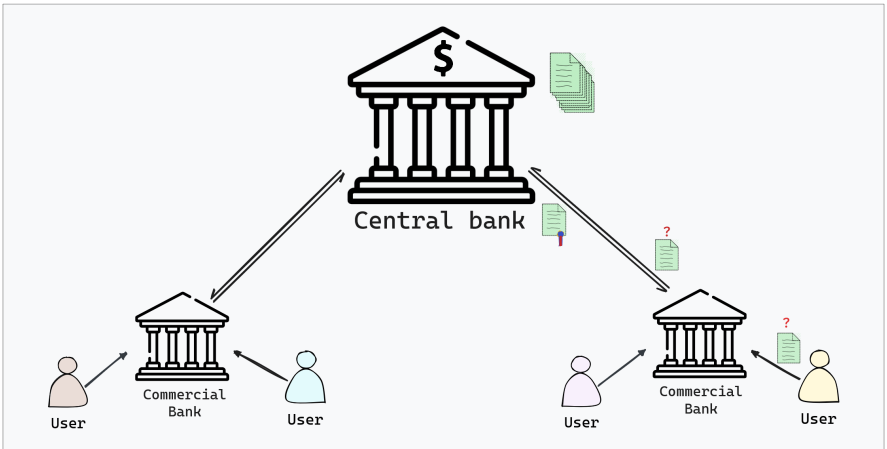


Figure 2.1: The hierarchical structure of a centralized financial ecosystem, where the central bank occupies the highest tier, exerting ultimate control over local commercial banks that form the intermediary layer. Users reside at the base and interact exclusively with local banks.

single entity. Without independent verification, users must rely entirely on the integrity of the central authority, increasing systemic risk and limiting accountability.

- Companies increasingly compete to gather user data, storing it in large data centers due to its high commercial value [45]. The challenge arises when data submitted to different services is aggregated under a single entity's control. Although users may share information willingly, such aggregation reveals personal information that users never intended to disclose. Their data and digital identity are managed by tech giants who construct comprehensive behavioral profiles that users cannot see, control, or correct [187]. Even when implementing end-to-end encryption, the central authority maintains visibility over critical metadata, including access patterns, request timing, and network relationships [158].

The limitations of centralized architectures motivated the emergence of distributed systems. Distributed systems break from single-machine execution by spreading computation across multiple nodes, offering varying degrees of decentralization and computational efficiency. At a first level, distribution primarily addresses computation. Workloads are partitioned and executed across multiple nodes to alleviate performance bottlenecks, enhance scalability, and reduce security vulnerabilities [32]. For instance, large language model training and inference, such as those employed by GPT-4 [2] and Gemini [3], utilize distributed computing clusters where model parameters and processing tasks are partitioned across thousands of GPUs, dramatically reducing training time from years to weeks or days [156, 169].

At a deeper level, distribution extends to governance and trust, eliminating centralized control entirely. This architecture results in fully decentralized architectures in which no single entity dictates system behavior. Contemporary distributed database systems such as the InterPlanetary File System (IPFS) [18], decentralized finance (DeFi) protocols [189], and decentralized digital identity frameworks like Self-Sovereign Identity (SSI) [141], and blockchain networks [143, 190] increasingly employ distributed governance models to promote transparency, auditability, and user autonomy.

[2] https://chatgpt.com/

[3] https://gemini.google.com/app

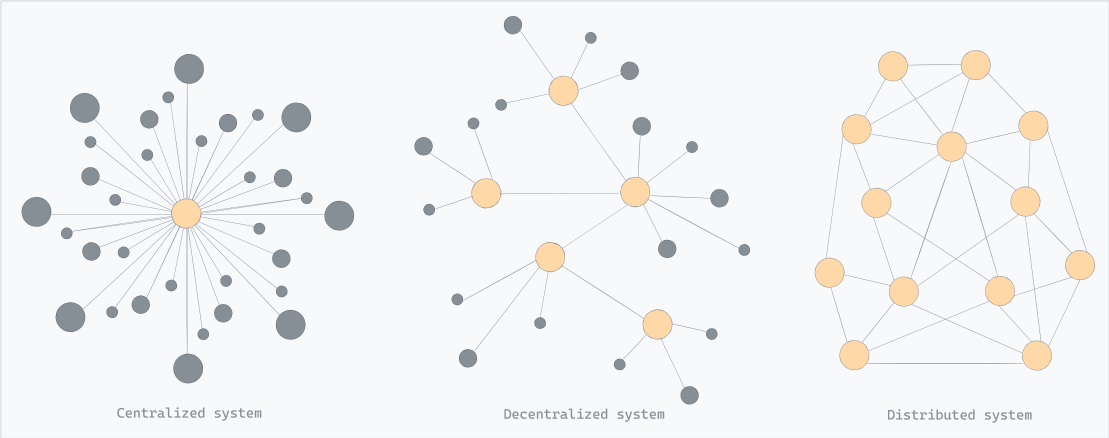


Figure 2.2: Centralized vs. distributed vs. decentralized system

The subsequent section explores how the foundations of distributed systems that enables the creation of resilient and trustworthy systems capable of addressing the challenges of security, transparency, and autonomy in increasingly complex computing environments.

2.2 Distributed Systems: Concepts and Challenges

2.2.1 What Are Distributed Systems?

A distributed system is a collection of physically independent processing units, commonly referred to as nodes, that interact by exchanging messages and collaborate to achieve a common goal, while appearing to users as a single coherent system [116]. Throughout the following chapters, we model a distributed system with the following. Let $P = p_1, p_2, \ldots, p_n$ denote a set of processes or nodes. A process is said to be correct if it follows the protocol specification, and faulty otherwise. We denote by f the maximum number of faulty processes. Each process p_i maintains a local state $state_i$ and communicates with other processes by sending and receiving messages over a network. Since processes do not share memory, they modify their states only through local computation and received messages. A transaction is denoted by tx, a block by B_i, and a ledger by $\mathscr{L}$. When a protocol produces an output, we say that a correct process decides a value v or delivers a message m, depending on the abstraction considered.

The architecture of a distributed system stands in contrast to centralized and decentralized models. As illustrated in Figure 2.2, the differences between these three system types become evident through their network topologies and node relationships. The centralized system shows star-shaped topology with a single central node, while the decentralized system exhibits scattered clusters of interconnected nodes without a central authority. The distributed system demonstrates a more complex mesh of interconnections where nodes maintain relationships across the network while potentially preserving some hierarchical structures.

A comparison of the core differences in control and architecture between these models are systematically summarized in Table 2.1. Specifically, while centralized systems offer simplicity of management, they suffer from a single point of failure. Decentralized systems excel in fault tolerance and transparency but can introduce complexity in governance and coordination. Distributed systems primarily provide scalability and

performance benefits through parallel processing and resource distribution, and can operate under either centralized or decentralized control paradigms depending on the specific implementation requirements [102].

Table 2.1: Comparison of Centralized, Decentralized, and Distributed Systems

Feature	Centralized	Decentralized	Distributed
Control	Single entity	Shared among peers	Both models
Trust Model	Central authority	Consensus mechanisms	Variable by design
Data Storage	Central database	Distributed ledger	Replicated across nodes
Scalability	Limited by bottleneck	High via distribution	High via replication
Fault Tolerance	Low	High	High
Transparency	Low – restricted access	High – open verification	Variable by design
Examples	Banks, social media	Blockchain, IPFS	Cloud services

A canonical implementation of distributed systems is the Peer-to-Peer (P2P) network, where nodes (*peers*) consuming and providing services simultaneously, create a flat network topology of equal peers that collaborate through direct interactions. Peers self-organize into an overlay network, forming a logical structure built on top of the physical internet. Rather than relying on dedicated servers, P2P topology enables each peer to contributes part of its own capacity to the network, including computational power for processing tasks, free disk space for storing files, and excess network bandwidth for data transmission [175].

The origins of P2P trace back to early file-sharing systems such as Napster (1999) and Gnutella (2000) [87], which demonstrated the potential of peers to exchange content without centralized servers. BitTorrent (2001) [43] refined this model by enabling efficient large-scale distribution through swarming techniques, and later platforms such as Skype leveraged P2P overlays for real-time communication [16]. More recently, P2P has provided the foundation for blockchain systems, where peers collaboratively maintain a distributed ledger without relying on central authorities [143, 190].

2.2.2 Key Characteristics of Distributed System

Distributed systems constitute a fundamental computing paradigm where independent components coordinate through message passing. These systems appear as unified entities while operating across physically separated nodes. The distribution of components introduces distinct properties that differentiate them from centralized architectures. On the one hand, distributed systems can offer fault tolerance by replicating functionality across nodes, scalability by parallelizing workloads, and privacy by avoiding single points of data concentration. On the other hand, distribution introduces hard challenges: ensuring concurrency control when multiple nodes operate simultaneously, maintaining consistency across independently evolving states, and achieving coordination in the absence of a global clock or shared memory [88]. In this section, we identify the fundamental characteristics that define distributed systems.

Replication

An important feature of distributed systems is data replication. Data is generally replicated to enhance reliability or improve performance. For example, if a file system has been replicated it may be possible to continue working after one replica crashes by simply switching to one of the other replicas. The other reason for replicating data is to improve performance. Performance replication is important when a distributed system needs to scale [175].

The principal challenge introduced by this redundancy is the maintenance of a coherent system state. Whenever a replica is updated, it becomes temporarily inconsistent with the others, creating a potential for conflicting views of the data. Coordination in distributed systems is often much more difficult compared to that in single processor or multiprocessor systems. Peers, which may join, leave, or fail at any time, cannot rely on direct trust relationships. Instead, they must establish trust through cryptographic protocols and distributed agreement mechanisms.

To propagate updates and manage membership in dynamic environments, protocols such as gossip algorithms [105] enable information to be propagated gradually and reliably across the network. In this process, a node periodically selects a few neighbors and exchanges the latest information it has, causing knowledge to spread organically like an epidemic propagation. At the same time, agreement protocols allow a network of mutually distrusting parties to agree on a common view of the system, even in the presence of random communication delays, failures and nodes joining and leaving the network [48]. These mechanisms collectively replace centralized enforcement with distributed validation.

Consistency

Consistency ensures that all nodes in a distributed system maintain a uniform view of data, even as updates propagate across replicas [175]. Achieving consistency depends critically on how operations are ordered. Lamport's seminal work on logical clocks [116] introduced a logical mechanism for assigning timestamps to events in distributed systems, ensuring a consistent order of events. Each process maintains a counter that increments with every local event. A partial order allows nodes to agree only on the ordering of causally related events, tolerating unrelated events to be unordered. In contrast, a total order can be constructed as a stronger form of ordering where every event is comparable to every other event [93].

Formally, Let E be a set of events. In distributed systems, Lamport's *happens-before* relation orders causally related events but may leave concurrent events unordered. A total order is a relation in which every pair of events is comparable. Thus, for any two events $e_1, e_2 \in E$, either $e_1 \rightarrow e_2$ or $e_2 \rightarrow e_1$. Partial order captures causality, while total order captures global sequencing of the events.

Consistency becomes challenging when nodes may receive and process updates asynchronously across varying network conditions. Several consistency models address these challenges, offering different guarantees, each representing distinct trade-offs between consistency strength, system availability, and performance. A consistency model specifies the values that read operations are allowed to return after write operations have occurred in a replicated system.

A system is *strongly consistent* if every read returns the result of the most recent completed write according to a single global order of operations.

A system is *eventually consistent* if, after updates stop and messages are eventually delivered, all correct replicas converge to the same value.

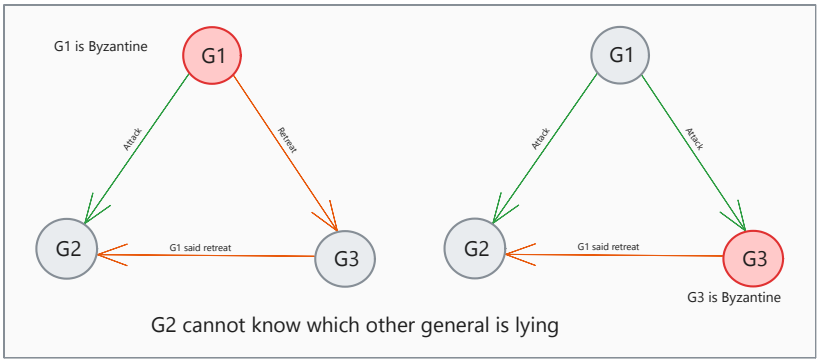


Figure 2.3: The Byzantine Generals Problem

Fault tolerance

Fault tolerance in distributed systems refers to the ability of the system to continue operating correctly even when some of its components fail [88]. Distributed systems necessarily operate under the assumption of various fault classes, including:

- **Crash failures:** A component abruptly ceases all operations and communication without warning or error signals. While conceptually simple, crash failures can introduce significant complications where components may need to distinguish between actual crashes and temporary network latency. The system may mistakenly continue to interact with the failed component, causing delays.
- **Network failures:** Network failures manifest as message losses, unpredictable delays, or network partitions that isolate groups of nodes from one another. Network failures can be transient or long-lasting and may affect system availability and consistency. The fundamental trade-off between these properties is formalized in the CAP theorem [78], which states that a distributed system cannot simultaneously guarantee consistency, availability, and partition tolerance. Consequently, systems implement mechanisms to detect and manage partitions while maintaining the guarantees of their chosen consistency model.
- **Byzantine failures:** Represent the most general and challenging category of failures, where components behave arbitrarily or maliciously. We refer to such components as *Byzantine*, a term initially introduced by Leslie Lamport in his seminal work "*The Byzantine Generals Problem*" [118], which has since been used to encapsulate all conceivable forms of failures a system can exhibit (See Fig. 2.3). A Byzantine faulty node might send conflicting information to different parts of the system or produce invalid results while appearing to function correctly. Systems requiring tolerance to Byzantine failures must implement more complex protocols than systems handling only crash failures[117].

2.2.3 The Consensus Problem

In a distributed system, the nodes collectively maintain a shared, coherent state. Challenges arise whenever a user proposes a state change, such as recording a transaction, updating a ledger, or modifying a system parameter. For the system to remain functional and reliable, all correct nodes must agree on the outcome of this change and its sequential order relative to other changes. To illustrate, in a distributed banking ledger,

when a user makes a deposit into their account. The deposit must be recorded to the account balance consistently across all replicas. To reach agreement, the replicas must exchange messages to ensure that the entire network have the same view of the account balance. If one node fails or behaves incorrectly, the other nodes must be able to detect this and continue to operate correctly. This scenario illustrates the *consensus problem*, a fundamental challenge in distributed computing.

Definition and Properties

Let $P = p_1, \ldots, p_n$ be a set of processes. Each process p_i proposes an input value v_i. A consensus protocol requires every correct process to eventually decide a value v, while satisfying: (1) *Agreement:* no two correct processes decide different values. (2) *Validity:* if a correct process decides v, then v must be a value proposed by some process. (3) *Termination.* every correct process eventually decides.

Consensus protocols are designed to guarantee *safety* (honest nodes reach a consistent state) and *liveness* (the system continues to progress) as long as adversarial nodes remain below a critical threshold [84]. While *termination* defines the liveness property associated with the consensus problem, the *agreement* and *validity* define its safety properties [39].

The concept of decentralized consensus lays the foundation for trustless environments, replacing reliance on any single entity with public verifiability. Decentralized consensus underpins trust-minimized systems by replacing reliance on any single actor with public verifiability. Common consensus mechanisms include Proof-of-Work (PoW) [143], Proof-of-Stake (PoS) [30], or Practical Byzantine Fault Tolerance (PBFT) [38], where participants achieve collective agreement through distinct approaches to participation and verification. The strength of consensus-based trust lies in its capacity to offer probabilistic or deterministic guarantees regarding system consistency and fault tolerance, even in the presence of faulty or adversarial participants. Though stated simple, reaching consensus may be difficult due to various reasons such as node crashes, malicious behavior, network faults, and latency arising from the absence of a global clock.

Impossibility results

The theoretical guarantees of consensus protocols, however compelling in formalized models, encounter substantial challenges when implemented in practical distributed systems. In synchronous environments, consensus is achievable even with arbitrary crash faults because we can make assumptions about the maximum time it takes for messages to get delivered [60]. However, asynchronous networks, characterized by unbounded message delays, impose severe limitations. The classical impossibility results of Fischer, Lynch, and Paterson [69] proves that no deterministic algorithm can guarantee both safety and liveness if even a single node might crash.

Further constraints were formalized by Lamport et al. in the *Byzantine Generals Problem* [118], proving that consensus is impossible if more than one third of the nodes are Byzantine in asynchronous networks. If more than two-thirds of all nodes in a system are honest, then consensus can be reached.

Additionally, in distributed systems, these synchronization primitives ensure that concurrent processes will not inadvertently overwrite each other's data. The power of a synchronization primitive in asynchronous distributed systems can be quantified using the consensus number concept introduced by Maurice Herlihy [92]. This theoretical measure indicates the maximum number of processes for which the primitive can solve the consensus problem in an asynchronous environment.

These theoretical constraints underscore that practical distributed systems, including blockchain implementations, often operate under relaxed assumptions about network synchrony, fault models, or the proportion of honest participants. Consequently, many systems adopt probabilistic or hybrid consensus approaches and synchronization primitives with lower consensus numbers that achieve eventual consistency and fault tolerance, even under adversarial conditions.

2.3 Blockchain: A Trustless Computational Model

The emergence of blockchain technology demonstrates practical solutions to achieving consensus in adversarial, decentralized networks where participants cannot assume trust in peers or intermediaries. It combines decades of research in cryptography, consensus algorithms, and peer-to-peer systems to create a robust, trustless framework. Blockchain provides a mechanism for mutually distrusting entities to maintain a synchronized, tamper-evident ledger without relying on centralized authorities. Bitcoin was the first successful cryptocurrency to leverage blockchain as its foundational data structure. Blockchain emerged as a solution to the double-spending problem in digital cash systems, using a replicated append-only ledger and a consensus mechanism to agree on a valid transaction history. The double-spending problem represents a critical flaw in digital currencies, where the same token could be spent multiple times without a central authority to track transactions.

Early blockchain implementations served primarily as distributed ledgers for cryptocurrencies (e.g., Bitcoin [143], Litecoin [126]). However, with the advent of Ethereum [28] in 2015, the capabilities of blockchain technology expanded beyond simple digital currency transactions. Ethereum architecture introduced the concept of a Turing-complete virtual machine, the Ethereum Virtual Machine (EVM), capable of executing arbitrary code on the blockchain [190]. The EVM functions as a distributed state machine that executes identical bytecode across all network nodes, ensuring consensus not only on transaction records but also on computation results.

This innovation enabled a higher abstraction layer through the implementation of smart contracts, which are a self-executing agreements with code directly written into the blockchain. Smart contracts facilitate the trustless execution of complex financial and organizational processes. Smart contracts subsequently marked a shift from the fat protocols stage, where all value is generated in the protocol layer, to the Decentralized Applications (Dapps) stage where transactions are programmable through smart contracts. In this new model, the blockchain transforms from processing simple transactions into general-purpose computing infrastructure capable of supporting the potential applications of blockchain technology beyond its initial use case as a digital currency system.

Despite their diverse implementations, the robustness of blockchain systems is underpinned by the integration of three core technological components:

- A distributed append-only ledger which allows new data records to be added at the end of the chain and prevents any alteration or removal once recorded. Each block contains a batch of transactions, a timestamp, and a cryptographic hash of the previous block. Such structure ensures data immutability and transparency, as every participant can verify the complete history of transactions.
- Cryptographic verification, including hash functions and digital signatures, that guarantee data integrity and authenticity. The security and integrity of blockchain data are maintained through advanced cryptographic techniques. Hash functions generate unique fixed-length outputs for any input data, creating unique digital fingerprints for each block. The hash values link blocks together,

forming a chain in which any alteration to a block would be immediately detectable. Additionally, digital signatures created using public-key cryptography, authenticate transaction origins and prevent unauthorized spending. Together, these cryptographic primitives guarantee that data remains consistent, tamper-evident, and verifiable by all network participants.

- Decentralized consensus algorithms that enable network-wide agreement on the system's state and the validity of operations. Protocols such as Proof-of-Work (PoW) or Proof-of-Stake (PoS) enable the network to reach agreement even in the presence of adversarial nodes. These algorithms are essential for securing the network, preventing double-spending, and ensuring that the blockchain continues to grow in a consistent manner.

2.3.1 Basic Blockchain Structure

Blockchain is a distributed ledger composed of a sequence of transactions grouped into cryptographically linked blocks. The stored data is maintained by participants in a peer-to-peer network executing a distributed consensus protocol to ensure the validity of the blockchain without relying on a third party or trust assumptions [140]. The network is composed of nodes that collaborate to maintain and update a distributed record of transactions.

The inherent heterogeneity of blockchain architectures poses a significant challenge to developing a unified mathematical model. Existing modeling approaches vary widely in scope and granularity, each tailored to capture the specific design choices of individual systems. As a result, no universal framework has yet emerged that can adequately represent all blockchain protocols.

For the purposes of this thesis, we adopt Garay's model [73], which was originally developed to analyze the Bitcoin blockchain protocol. Garay et al. proposed a formal abstraction that covers the core components of the Bitcoin protocol into a mathematical framework, enabling rigorous security analysis. Despite its original focus on Bitcoin, we contend that this model provides sufficient generality to represent fundamental elements common to multiple blockchain architectures.

Formally, following Garay's model [73], a blockchain can be defined as a public chain of an ordered sequence of blocks: $\mathscr{L} = (B_1, B_2, \ldots, B_n)$. The ledger $\mathscr{L}$ is extended to a new longer chain $\mathscr{L}' = \mathscr{L}||B_{n+1}$ by attaching a (valid) block B_j. Each block B_i is defined as the tuple $B_i =< s_i, x_i, ctr_i >$ with $i \in \{0, n\}$.

- $s_i \in \{0,1\}^\kappa$ refers to the hash of the previous block of size κ bits.
- $x_i \in \{0,1\}^*$ is the binary sequence that represents data
- $ctr_i \in \mathbb{N}$ is the *nonce*, a counter used for the proof-of-work algorithm.

The hash on the block header identifies each block in the blockchain. It also refers to a previous block linking each block in the blockchain to its predecessor. The sequence of hashes connecting each block to its parent forms a chain that extends all the way back to the *genesis* block (the first block of the chain) as illustrated by Fig. 2.4. The blocks B_0 and B_n are respectively the *genesis* block and the latest block. We note that for the genesis block B_0 there is no previous block to reference. Therefore, B_0 components (i.e., ctr_{-1} and s_{-1}) are typically hardcoded directly into the blockchain.

Transactions in a block are compressed into a single hash called the Merkle root. A Merkle root is built by recursively hashing pairs of transactions until there is only one hash, known as the Merkle

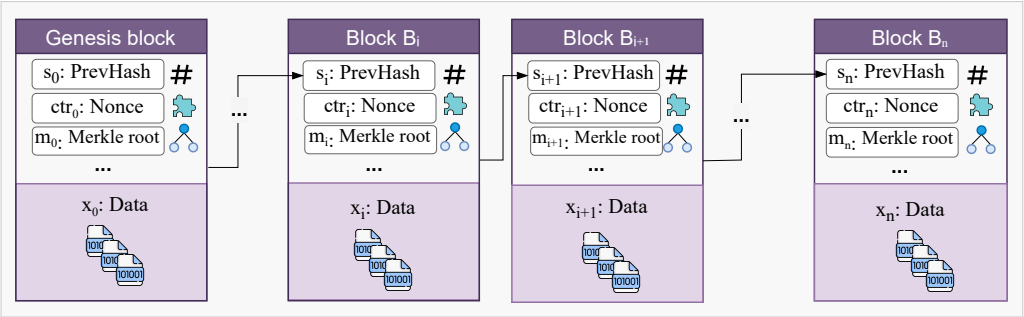


Figure 2.4: Blockchain ledger components

root. Formally, for a block B_i, the Merkle root m_i is computed using a collision-resistant hash function $G : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ where $m_i = G(tx_1, tx_2, \ldots, tx_n)$. The recurrent hashing process represents a compact summary of transactions and enables verification of transactions without the need for a complete local copy of all transactions. Therefore, any transaction tampering would produce different hash values which would be detected.

When a node initiates a transaction, it is digitally signed using the private key of the sender. Likewise, the corresponding public key of the sender is used by other nodes to verify the authenticity of the signature. Once signed, the transaction is broadcast to the network and enters a transaction pool (or mempool), where it awaits validation. Specialized nodes with significant computational resources, often referred to as miners, select transactions from the mempool and validate each one by checking digital signatures and compliance with requirements set by the application. The miners aggregate validated transactions into a block and compete with other miners to append to the blockchain through a consensus mechanism.

To create a valid block, a miner must add a random number, known as a nonce, to the header of the block such that the resulting hash value to be below a predefined difficulty level. Formally, a block B_i is valid *iff* the following equation holds:

$$ValidBlock_q^D(B_i) := (H(ctr_i, G(s_i, x_i)) < D) \wedge (ctr_i < q) = 1 \tag{2.1}$$

With the parameter D being the current block difficulty level, and q is the maximum number of hashing attempts permitted in a mining round. The difficulty parameter is periodically adjusted to maintain consistent block production intervals despite fluctuations in network hash rate. Fig. 2.5 depicts an illustration of the Proof of Work algorithm, demonstrating how miners search for valid blocks by satisfying the cryptographic difficulty constraint.

Once a miner finds a valid nonce ctr_i, the block B_i is broadcast to the network. Other nodes verify its validity by checking: i) The correctness of the nonce ctr_i. ii) The integrity of transactions via the Merkle root m_i. iii) The backward hash link $s_i = H(B_{i-1})$. If valid, the nodes adopt the new chain $\mathscr{L}' = \mathscr{L}||B_i$.

2.3.2 Core Properties

Besides the common features that blockchain inherently possesses as a distributed system, blockchain systems are characterized by particular characteristics that vary depending on their implementation, topology, consensus mechanism, and application domain. We articulate the most relevant features in the context of

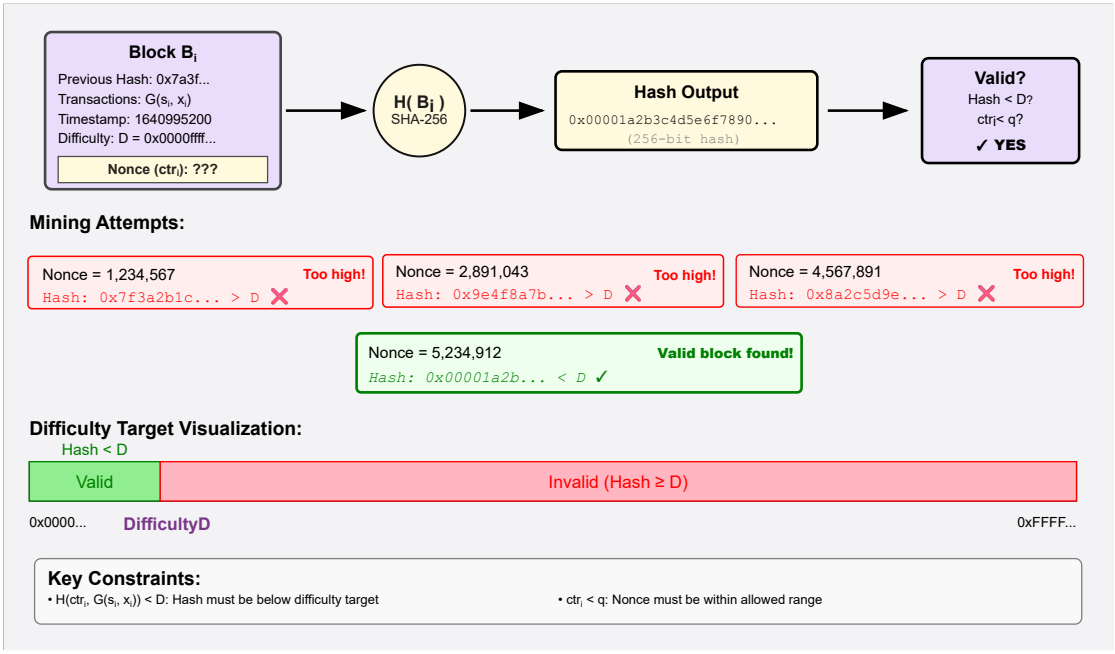


Figure 2.5: Proof of Work Mining Process. Miners iteratively test different nonce values (ctr_i) until finding one that produces a hash below the difficulty target D. The difficulty visualization shows the hash space partition, where only hashes falling below the current difficulty target D (highlighted in green) are considered valid. The constraints ensure both cryptographic proof of work (hash below difficulty) and computational limits (nonce within allowed range of hash queries in a round q).

this thesis in the following:

- **Immutability.** Immutability refers to the hardness of rewriting data on the chain after it has been added. It enables network participants to be confident that transactions on the chain cannot be altered, thereby preventing double-spending attacks. The chain immutability is primarily maintained through the cryptographic chaining of blocks of data. Each block contains a hash of the previous block, forming a continuous chain extending back to the genesis block. Any attempt to tamper with data in a previous block would invalidate the hash of all subsequent blocks in the chain requiring an attacker to regenerate all succeeding blocks. The probability of an adversary successfully altering a block decreases exponentially with confirmation depth, making deep history practically immutable [143]. Given the computational cost associated with recomputing these hashes and achieving consensus in most blockchain systems, modifying the blockchain's history becomes computationally infeasible for all practical purposes.
- **Permission Models.** The degree of decentralization in a blockchain network is largely determined by its permission model, which dictates who can participate in the network. *Permissionless* blockchains represent the most decentralized implementation, operating as open networks where any entity can join, validate transactions, and participate in consensus mechanisms without prior authorization or identity verification. The openness ensures that no single entity controls the network, promoting transparency and resistance to censorship. To maintain security in this open environment, these systems implement Sybil-resistance mechanisms (such as PoW or PoS) that require participants to commit computational resources or financial stakes as a means to establish legitimate identities and prevent attack vectors. While maximizing trustlessness, this high level of decentralization

can lead to challenges in scalability and performance due to the coordination complexity among untrusted participants. In contrast, *permissioned* blockchains restrict participation to a predefined set of entities, often within a consortium or organization. While this model enhances control, efficiency, and compliance with regulatory requirements, it introduces elements of centralization, as the governing body has authority over the network. This structure can limit transparency and make the system more susceptible to censorship.

- **Transparency.** The ledger contains a full transaction history. As the blockchain is an open file, anyone can access it and audit transactions. The distributed ledger is typically replicated across all full nodes in the network, providing open verifiability of all transactions.

  Most blockchain systems, especially permissionless blockchains like Bitcoin and Ethereum, are characterized by a high degree of transparency. The distributed ledger is often publicly accessible, meaning that the entire transaction history, from the genesis block onwards can be viewed by any participant. The transparency allows any participant to independently audit the entire transaction history without requiring trust in a central authority. The transparency enhances accountability and allows for independent verification of the system's state and transaction history, fostering trust through verifiable openness.

- **Security.** This property depends on achieving agreement in a distributed setting, highlighting the critical role of consensus mechanisms. While the core properties define what blockchain systems achieve, consensus mechanisms determine how correct nodes converge on a common ledger state despite faults, adversarial behavior, and network uncertainty. The formal guarantees of these properties directly depend on the consensus protocol's ability to ensure that all honest participants converge on an identical view of the blockchain despite potential attacks and network partitions.

2.3.3 Consensus Protocols

Classical Byzantine Fault Tolerant Consensus.

Practical Byzantine Fault Tolerance (PBFT) [38] is a consensus algorithm specifically designed to address the Byzantine Generals Problem, which involves reaching consensus in a distributed system where malicious actors can inject false information. PBFT was designed to work efficiently in asynchronous (no upper bound on when the response to the request will be received) systems. At its core, PBFT assumes a system of multiple replicas, one of which is designated as the primary (or leader) and the rest as backups. The protocol ensures correct operation as long as fewer than one-third of the nodes are faulty, meaning in a system of $3f+1$ replicas, up to f can behave incorrectly without compromising overall safety. The core of the protocol of PBFT is composed of three phases, as illustrated in Fig. 2.6: The PBFT protocol operates in a multi-round voting process, where nodes take turns acting as the primary node (leader) responsible for proposing the next block of transactions. The key steps in the PBFT process include:

1. Request: A client sends a request to the primary node, which then broadcasts the request to all other nodes in the network.

2. Pre-prepare: Upon receiving the request, the leader broadcasts a message containing the proposed transaction to all other nodes, who then verify the validity of the request.

3. Prepare: After verifying the request, the nodes send a prepare message to indicate their readiness to commit to the proposed transaction.

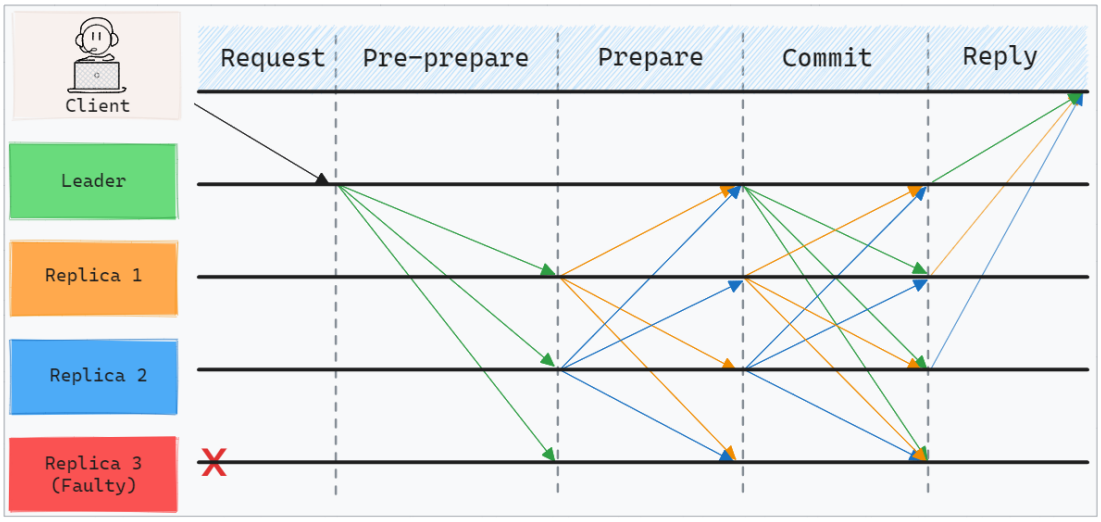


Figure 2.6: PBFT protocol workflow. When the primary receives a client request, it starts a three-phase protocol to atomically broadcast the request to the replicas. The three main phases are pre-prepare, prepare, and commit. The pre-prepare and prepare phases are used to totally order requests sent in the same view even when the primary, which proposes the ordering of requests, is faulty. The prepare and commit phases are used to ensure that requests that commit are totally ordered across views. Finally, the replicas wait for a sufficient number of commit messages before considering the transaction as confirmed, then the system issues the reply to the client.

4. Commit: Once a node receives prepare messages from at least a threshold number of nodes, it sends a commit message. Once a node receives the commit messages from a threshold number of nodes, it considers the transaction as committed.
5. Reply: Finally, the committed transaction is executed, and the response is sent back to the client who initiated the request.

PBFT ensures that all honest nodes agree on the validity and ordering of transactions. It provides safety, liveness, and finality guarantees, making it suitable for use cases where fast transaction confirmation is crucial. Despite its robustness, PBFT exhibits limitations in scalability and communication overhead, as its message complexity grows quadratically with the number of participating nodes ($O(n^2)$). Furthermore, frequent leader rotation and view-change procedures introduce latency and coordination costs that hinder large-scale deployment. To address these limitations, several optimized Byzantine Fault Tolerant protocols have been proposed, such as Tendermint [114], HotStuff [196], and HoneyBadgerBFT [139], which adapt the communication process, improve leader election efficiency, and enhance throughput under partial synchrony.

Proof-Based Consensus Protocols

Proof-based consensus protocols are a family of consensus mechanisms in blockchain systems where participants prove they have expended specific resources to gain the right to propose or validate blocks. These protocols establish Sybil resistance in permissionless environments by requiring verifiable commitment of resources before participation. Unlike traditional Byzantine fault tolerance mechanisms that depend on known validator sets [38, 24], proof-based approaches enable open participation while maintaining

probabilistic security guarantees proportional to resource investment. The fundamental principle of proof-based consensus is to create a barrier for adversaries, requiring a huge investment of resources that makes network manipulation prohibitively expensive. Participants must demonstrate their commitment through one of several resource-driven approaches. In the following, we list and briefly introduce the widely used Proof-based algorithms.

Proof-of-Work (PoW). As the pioneering consensus algorithm introduced in the Bitcoin whitepaper [143], PoW establishes decentralized consensus in an adversarial environment by leveraging a cryptographic computations and economic incentives. The PoW consensus requires network participants (miners) to compete in generating a verifiable proof of expended computational energy to propose a new block. Miners compete to find a *nonce* such that the cryptographic hash of a proposed block's header falls below a dynamically adjusted difficulty target (see Fig. 2.5). This process, known as mining, is computationally intensive by design, but trivial for any network node to verify. The first miner to find a valid proof broadcasts the new block to the network, receiving a block reward comprising both the fixed issuance and transaction fees. Upon reception, nodes independently validate the block's transactions and the correctness of the provided proof before appending it to their local blockchain. Security in PoW is probabilistic and derived from incentives and computational hardness. Temporary forks occur naturally when multiple miners produce valid blocks concurrently, resulting in transient parallel chains. Consensus eventually converges to the longest valid chain, defined as the one with the greatest cumulative work, with the probability of reversal diminishing rapidly with confirmation depth [73, 76]. Several proposals introduced optimizations to the mining process, adjusting difficulty, timing, or reward structures to improve resource efficiency. Examples include GHOST [171], Bitcoin-NG [67], and Proof-of-Useful-Work [14]. Despite these advances, PoW remains criticized for its high energy consumption and limited scalability, motivating the exploration of alternative consensus mechanisms.

Proof-of-Stake (PoS). The PoS [33] emerged as an energy-efficient alternative to the resource-intensive PoW. In PoS systems, participants (referred to as validators) lock a portion of their token holdings as a stake to gain eligibility for proposing and validating blocks. The stake determines the eligibility to participate in consensus. The probability of a validator being selected to propose or validate new blocks is typically proportional to the size of their stake, determined through a pseudo-random, stake-weighted selection algorithm. Under PoS, validators are incentivized to act honestly since any attempt at malicious behavior can result in severe penalties, including the loss of a portion or all of their staked tokens. Although PoS substantially improves energy efficiency and transaction throughput compared to PoW, it also redefines the foundation of security from computational effort to economic participation. This shift raises broader questions about how stake distribution, validator incentives, and governance affect decentralization and resilience over time. As a result, various PoS variants, such as Delegated Proof-of-Stake (DPoS) [119], Ouroboros [107] and Algorand [77], have been proposed to strengthen validator accountability, randomness generation, and network finality.

Novel Alternatives

Non-proof-based consensus protocols offer alternative strategies for achieving agreement in distributed systems, moving away from the resource-intensive nature of proof-based consensus protocols. These protocols prioritize emergent consensus, where participants collaboratively determine ledger state through iterative, decentralized voting or referential structures to ensure that participants agree on the state of the ledger.

A prominent category is Directed Acyclic Graph (DAG), as implemented in systems such as IOTA [152].

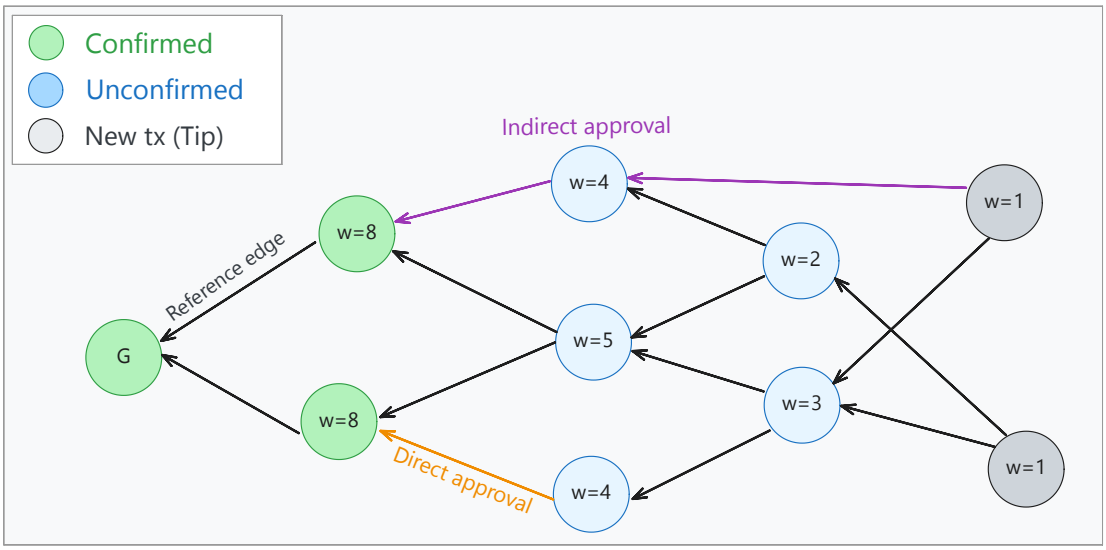


Figure 2.7: DAG-based consensus structure. Nodes represent transactions and edges denote approvals. New transactions concurrently reference multiple existing ones, increasing cumulative weight (W). Confirmed transactions (green) have high weight from extensive approvals, while unconfirmed ones (blue) await sufficient support.

Unlike conventional blockchain architectures, which enforce a sequential chain of blocks, DAG-based consensus structures transactions in a graph topology, enabling concurrent transaction validation and significantly improving scalability and throughput. In DAG-based systems, each new transaction must refer and approve multiple preceding transactions, forming a directed acyclic structure. Transactions are collectively validated by the network, using a cumulative weight metric which quantifies the level of validation each transaction receives. The cumulative weight metric $W(tx)$ for a transaction tx is computed recursively as the sum of the weights of all transactions that directly or indirectly approve tx. Over time, transactions with higher $W(tx)$ are considered confirmed, as they are embedded deeper in the DAG structure. A simplified diagram of DAG-based consensus is depicted in Fig. 2.7. The DAG structure allows for parallel validation and incorporation of transactions, potentially overcoming performance limitations inherent in sequential blockchain designs.

Another innovative direction is represented by the *Avalanche* protocol family [159], which departs fundamentally from both proof-based and leader-based paradigms. *Avalanche* achieves consensus through repeated randomized sampling and metastability, enabling rapid probabilistic convergence on conflicting transactions.

In an *Avalanche* network, each node repeatedly polls a small, randomly selected subset of peers to query their current preferences on a transaction. Based on the responses, a node updates its own preference if a supermajority, defined by a threshold (α), of the sampled peers agrees on a single choice; otherwise, it retains its current view. To ensure robustness, each node maintains a confidence counter that increases with every consecutive round where its current preference receives α-majority support from polled peers. The node finalizes its preference once this counter reaches a sufficient threshold. However, if a competing choice consistently achieves α-majority support, the node abandons its current preference, resets the counter, and begins building confidence in the new choice.

These sampling and update steps are performed iteratively across the network, and through successive rounds, local preferences quickly converge to a consistent state, ensuring that all honest nodes agree on the state of the ledger. Once a sufficient majority forms around a decision, metastability ensures that the probability of reversal becomes exponentially small. The *Avalanche* family builds upon this principle with hierarchical voting and confidence accumulation, achieving consensus with low communication complexity, high throughput, and strong resilience to Byzantine faults.

While these novel paradigms demonstrate significant potential, their evolution involves addressing open research questions concerning global consistency, fairness in leader or committee selection, and formal safety proofs under dynamic network conditions. Nevertheless, they mark a significant shift toward lightweight, scalable, and energy-efficient consensus models that transcend the limitations of proof-based and classical Byzantine approaches. These alternatives embody an ongoing research effort actively exploring how to optimally balance the blockchain trilemma that continues to shape the design of next-generation distributed ledgers.

2.4 Fundamental Blockchain Challenges

The blockchain trilemma is a concept that highlights the inherent challenge in achieving the three critical aspects of blockchain technology simultaneously namely, security, scalability, and decentralization. These three aspects are interdependent in such a way that the improvement of one often leads to the degradation of the other. In practice, blockchain systems must balance these conflicting priorities, leading to trade-offs that impact performance, energy efficiency, and user trust. The following subsections detail key challenges facing blockchain networks.

2.4.1 Computational Complexity

Blockchain consensus protocols impose significant computational burdens to maintain security and decentralization. Proof-of-Work (PoW) systems, for instance, require miners to solve cryptographic puzzles through brute-force hashing, consuming vast energy resources Bitcoin's annualized energy usage rivals that of small nations [21].

Even energy-efficient alternatives like Proof-of-Stake (PoS), demand complex cryptographic operations (e.g., verifiable random functions for validator selection [77, 107]) and constant signature verification for block validation. This computational race centralizes the mining power among entities with access to cheap energy. Hence, incentivize collusion in large computational pools and undermine decentralization [74]. Even lightweight nodes, which rely on Simplified Payment Verification (SPV), inherit security dependencies on full nodes, creating a trust asymmetry [143, 76]. On the other hand, Layer-2 solutions (e.g., rollups [177], state channels [151]) and sharding [31, 33] (e.g., Ethereum 2.0) mitigate computational overhead by offloading transactions from the base layer, they introduce new complexities in cross-shard communication and tamper-proof verification. These trade-offs underscore the challenge of balancing security, decentralization, and efficiency in blockchain design.

2.4.2 Privacy Considerations

Transactions on public blockchains remain visible to all participants, creating significant challenges for applications that require confidentiality. Although identities are represented by cryptographic addresses, the

transaction history associated with these addresses is fully transparent. Despite pseudonymous addresses, blockchain transparency exposes transaction patterns and metadata to public scrutiny. Privacy-focused chains like Monero (ring signatures) [181] and Zcash (zk-SNARK) [163] employ cryptographic primitives to obscure transaction details, but such mechanisms often incur computational overhead or require trusted setups, thus affecting centralization and energy for privacy. Furthermore, privacy enhancements conflict with regulatory mandates for anti-money laundering compliance, creating tension between user anonymity and legal accountability. More recent approaches, including zero-knowledge proofs with selective disclosure [162], aim to mitigate this tension by enabling users to reveal only specific attributes when necessary while preserving overall transactional privacy.

2.4.3 Immutability Constraints

Immutability is a cornerstone of the blockchain trust model that ensures a tamper-resistant record by cryptographically binding blocks into an append-only ledger. However, this rigidity complicates error correction, data removal (e.g., for GDPR compliance [75]), and responses to malicious content (e.g., illegal data stored on-chain). For instance, Ethereum's contentious 2016 hard fork to reverse the DAO hack exemplifies the ethical and technical dilemmas of overriding immutability. Thus, blockchain systems must navigate the dual imperatives of preserving integrity while accommodating legitimate revisions.

2.4.4 Storage and Performance

Blockchain's append-only data structure creates monotonically increasing storage requirements. As the blockchain grows, scalability bottlenecks emerge. The exponential growth in chain size creates significant storage challenges for network participants. As of March 2025, the Bitcoin blockchain exceeds 600GB, while Ethereum's state size surpasses 1TB when including historical data [20]. This growth trajectory presents significant challenges for node operators and threatens decentralization by raising hardware requirements. The technical issue stems from the immutability property that prevents pruning of historical data without compromising the system's security model.

Blockchain's decentralized storage model requires each node to maintain a full copy of the ledger, leading to scalability bottlenecks as chain size grows exponentially. While pruning techniques [134, 146] (e.g., retaining only Unspent Transaction Output (UTXO) sets) and decentralized storage solutions alleviate storage demands, they compromise data availability or introduce external dependencies. Performance limitations further constrain throughput: Bitcoin processes 7 tx/s, compared to Visa's 65,000 tx/s. Layer-1 optimizations and Layer-2 networks improve throughput but often sacrifice decentralization or finality guarantees. These challenges represent the blockchain trilemma, where gains in scalability typically come at the expense of security or decentralization.

2.5 Conclusion

This chapter establishes the technical foundations necessary to understand the evolution from centralized architectures to decentralized blockchain systems. Centralization emerged as a fundamental constraint that concentrates power, creates vulnerability of a single point of failure, and enables censorship. While distributed systems emerged as a response, offering key principles like concurrency, consistency, and fault tolerance, they remain constrained by the difficulty of establishing trust in adversarial environments.

Legacy approaches, including classical consensus protocols such as PBFT, depend on fixed or identifiable participants, which inherently excludes open, permissionless participation.

Blockchain technology resolved the trust problem through cryptographic guarantees and distributed consensus, enabling coordination among anonymous participants without central authority. However, the mechanisms that provide security and decentralization impose substantial costs. Distributed consensus requires extensive message exchange. Immutable storage accumulates unbounded data. Public verification exposes transaction details. These inherent tensions manifest as the blockchain trilemma captures the difficulty of simultaneously achieving scalability, security, and decentralization.

The rest of this thesis builds on these insights. Subsequent chapters tackle these structural limitations by introducing lightweight consensus mechanisms for decentralized value exchange, designing scalable blockchain architectures suited for heterogeneous and resource-constrained environments, and proposing privacy-preserving models that enhance user data control. These contributions target the core barriers preventing blockchain deployment in resource-constrained and privacy-sensitive environments, advancing toward genuinely inclusive decentralized systems.

3 Consensusless Transfers: A Shift from PoW

The foundational premise of blockchain technology has long relied on consensus protocols to resolve conflicts, prevent double-spending and Sybil attacks, and maintain global state consistency. The evolution of consensus mechanisms, from the computationally intensive PoW to more energy-efficient alternatives like Proof-of-Stake (PoS), has yielded diverse approaches to distributed agreement. Nevertheless, the *blockchain trilemma* (decentralization, scalability, security) reveals that existing solutions optimize for at most two dimensions of the trilemma, while invariably compromising on the third. Such persistent constraint suggests that the limitation may not lie in the particular implementation of consensus, but rather in the fundamental nature of the consensus paradigm itself. A critical question arises: *Is consensus, as traditionally formulated, an inviolable requirement for distributed transaction systems?*

In response to these limitations, the concept of "*consensusless*" transaction transfer has emerged as a promising alternative. In a consensusless system, instead of requiring nodes to agree on a total order, the system can directly establish eventual consistency for transactions without running a resource-intensive consensus algorithm.

In this chapter, we introduce LIGHTX a consensusless transaction systems, which forgo explicit agreement phases in favor of probabilistic guarantees, enabling scalable, lightweight transfers without sacrificing Byzantine fault tolerance. LIGHTX represents an efficient implementation of a consensusless blockchain protocol. It is designed not only to encapsulate these theoretical advantages, but also to validate its performance under realistic network conditions.

Empirical evaluations indicate that systems employing probabilistic broadcast and lightweight resource-testing mechanisms, such as those based on Proof-of-Bandwidth (PoB), can achieve near-instant finality without sacrificing security against adversarial behavior. Moreover, the drive for scalability and integration with devices having limited resources necessitates a paradigm that minimizes energy consumption and computational complexity. The lightweight design inherent in consensusless protocols not only addresses the environmental and economic drawbacks of PoW but also facilitates deployment on heterogeneous networks, where traditional consensus is impractical.

3.1 Related Work

Despite being a major asset, PoW by its very nature incorporates a serious handicap. The excessive energy consumption induced by mining activity and the time required for transaction validation result in high costs and low scalability. PoW systems naturally develop a concentration of power centered around miners, resulting in an ecosystem where only participants possessing massive computing power can participate in the consensus process. Lately, new families of consensusless algorithms are emerging. AT2 [83] proposes multiple protocols for assets transfer based on Scalable Byzantine Reliable Broadcast (sBRB) abstraction to build an asset transfer system abandoning the expenses of consensus. AT2 also refers to the idea of a *Proof-of-Bandwidth* protocol but does not describe any comprehensive practical solution. [159] combines traditional consensus and Nakamoto's consensus [143] to come up with a cost-effective protocol that makes use of gossip protocols with recurrent sampling. [159] however, has a communication complexity of $O(kn)$ where $(k << n)$ is a security parameter.

Other alternatives opting for randomization to achieve agreement, such as HoneyBadgerBFT [139] and BEAT [59], stress on the security aspect by setting extensive use of cryptographic operations (i.e., erasure coding and threshold encryption). They manage indeed to reduce communication complexity (linear at the best case) and enhance system security. However, they result in a non-negligible computation overhead. Algorand [77] is also a recent cryptocurrency that confirms transactions with latency on the order of a minute while scaling to thousands of users. Algorand uses a new Byzantine agreement protocol to reach consensus. The consensus protocol used by Algorand consists of a synchronous protocol that incorporates a PoS system with a Byzantine fault tolerance agreement.

Solutions such as off-chain blockchain [151] have emerged to solve the scalability problem. The Lightning Network is an off-chain [151] that establishes payment channels for micro-payments that do not require to be included in the blockchain one by one. Consequently, payment networks can achieve higher transaction throughput and lower latency than blockchain. However, they also suffer from security challenges and impractical trust assumptions. None of the aforementioned achieve consensus in logarithmic per-node communication cost and provide a low computation and communication overhead when dealing with the threat enforced by Sybil attack.

3.2 System Model

We consider a public asynchronous fully connected message-passing system that provides direct communication between each pair of nodes. The term node and process are used interchangeably. The system builds upon the following assumptions:

Nodes. Nodes are denoted $p_1, .. , p_n$. We assume that each node is associated to an account and each account has a single owner. We identify a transaction issued by p_i to p_j as a transfer of a digital asset from the account of p_i to the account of p_j. We consider a straightforward probability sampling strategy, by deploying a sampling oracle Ω that, at each execution, generates randomly selected samples (i.e. subsets with parametrized sizes) of nodes among the total population of nodes. All nodes have direct access to the sampling service to pick their samples. We leverage existing encryption and authentication primitives: digital signature, SSL certificates and hash functions, which are known to be hard to break. Then we assume that adversaries cannot break cryptographic primitives.

Links. We consider authenticated reliable point-to-point communication links [34]. In principle, reliable links guarantee delivery and no message duplication. Given any message m sent from a node p_i to a node p_j, m is delivered to p_j exactly once. Authentication is deployed to ensure no creation property (i.e., to prevent malicious nodes from creating and propagating a message and pretending it originates from the original sender). Such a link abstraction is implemented by lower-level transport protocols.

Churn. Participating nodes are recommended to dedicate all their bandwidth resources for the application, especially during assessment phase. This is due to the fact that the bandwidth will be shared arbitrarily between the set of running applications on their devices. Consequently, it results in undesired fluctuated measurements caused by this bandwidth usage. Thus, we can track the bandwidth changes and assert that they reflect the behavior of the nodes in the application. We also assume that nodes have bounded bandwidth resources, but they are not required to be homogeneous in terms of bandwidth capacity, since the system only relies on detecting bandwidth fluctuations and not the bandwidth capacity. Nodes newly joining the network are not assigned any reputation score; reputation scores are only assigned after interacting with new nodes and recording their bandwidth measurements. New-comers (includes recently active nodes) get their scores either by referring to previously calculated global scores (measured by the network) or by proceeding to a new PoB phase.

We also assume a set of high-ranked *pre-trusted*nodes, that a node can trust their opinion about the network status. The *pre-trusted* set can be chosen on different bases. For instance, choosing the first nodes joining the network [103], or dynamically choosing highly reputable nodes based on their history [203, 113]. We opt for a hybrid approach to assess bandwidth metric via both proactive and reactive assessment. First, proactively, by launching the *PoB* mechanism periodically (The period τ is a system parameter that is determined at application level). And reactively, by responding to the network changes, in terms of network size and transaction submission rate. During a measurement phase, we consider the system to be of static size.

Threat model. We consider that nodes may experience Byzantine failures (i.e. nodes fail in an arbitrary manner that may deviate from the algorithm [34]). Byzantine nodes are represented by f computationally unbounded malicious nodes that may collude to interfere with the algorithm. We also assume that Byzantine nodes cannot compromise authentication primitives.
Scenario (1): We define a Byzantine node as a node that acts maliciously by presenting conflicting or inconsistent messages with the objective to carry out a double-spending. Double-spending occurs when a transaction uses the same input as another transaction that has already been spent. Concretely, the transaction is spent twice.
Scenario (2): We consider Sybil adversaries who declare multiple fake identities in the network and thus gain a majority opinion to send out inconsistent transactions. A Sybil attacker may also collude with other Byzantine nodes on the network, who may falsely report each other's scores. When dealing with powerful adversaries (e.g., nationwide or global adversaries), we need to deploy more advanced probing techniques that enable nodes to quantify the dedicated bandwidth resources of their peers and thus detect fluctuations caused by adversarial behavior. At the current stage, the analysis of appropriate probing techniques has not yet been studied.

3.3 Background and Technical Foundations

Consensus protocols constitute a core component in blockchain systems, as they define how agreement is reached among distributed participants, especially in the presence of faults or adversarial behavior. The

choice of consensus mechanism directly affects the scalability, security, and efficiency of the overall system. In this section, we review two models of consensus and the guarantees offered by both deterministic and probabilistic consensus and then we give an overview of scalable Byzantine Reliable Broadcast primitive and mention its limitations.

3.3.1 Deterministic vs. Probabilistic Consensus

Deterministic consensus protocols, such as Practical Byzantine Fault Tolerance (PBFT) [38] and Hotstuff [196], provide finality guarantees. Every honest participant in the system will eventually agree on the exact same outcome. There is no possibility of conflicting valid states once consensus is reached. In such protocols, agreement is reached through a structured voting or communication process where nodes explicitly exchange messages and come to a definitive conclusion.

On the other hand, deterministic algorithms are restricted by impossibility results [69]. The FLP impossibility result imposes that, in asynchronous environments, no deterministic protocol can simultaneously guarantee termination and correctness in the presence of even a single faulty node. More constraints state a lower bound limit in the presence of faulty processes and asynchrony assumptions [118]. In a nutshell, deterministic algorithms often have a stricter limit on the number of faulty or malicious nodes they can tolerate to guarantee consensus. These protocols typically operate under partially synchronous network assumptions and can tolerate up to f Byzantine failures in a system of $3f+1$ nodes. Additionally, deterministic approaches often struggle with scalability issues as the number of participating nodes increases, primarily due to their quadratic communication complexity $O(n^2)$.

In contrast an alternative that makes it possible to achieve consensus is to enable randomization. Probabilistic consensus ensures rather probabilistic *liveness* properties (every operation eventually completes), but guarantees the same *safety* properties (nothing wrong happens) [7].

Probabilistic consensus protocols like Nakamoto consensus [143] and its derivatives provide eventual consistency with increasing finality over time. In many of these systems, particularly proof-based protocols, participants must commit scarce resources to gain the right to propose or validate blocks. This resource commitment serves as a Sybil-resistance mechanism, ensuring that influence in the network is proportional to some verifiable cost. Examples include Proof-of-Work (PoW) [143], where nodes solve computational puzzles, Proof-of-Stake (PoS) [33, 107], where validators are selected based on staked assets, and other models leveraging storage or bandwidth, such as Proof-of-Space and Proof-of-Capacity [62, 148, 44].

These protocols scale to thousands of nodes by accepting probabilistic finality, where multiple confirmations gradually increase security. However, this scalability often comes at the cost of physical resource commitments such as computation, stake, or storage. The probabilistic nature introduces variable latency and potential temporary forks, making them less suitable for applications requiring strong consistency guarantees. Randomized algorithms, however, are often simpler, provably faster, and offer better defense against adversarial behavior [111].

3.3.2 Scalable Byzantine Reliable Broadcast Primitive

Byzantine Reliable Broadcast (BRB) abstraction [34] represents a fundamental primitive in distributed systems that provides weaker guarantees than full consensus but offers significant efficiency advantages in certain contexts. Standard Byzantine Reliable Broadcast (BRB) ensures that if a correct sender broadcasts

a message, all correct nodes deliver that message, and if a Byzantine process broadcasts different messages to different nodes, either all correct processes deliver the same message or none do. In real terms, existing broadcast protocols make use of quorum-based techniques. Conceptually, quorum schemes require the confirmation of the majority of participants to deliver a message (i.e., the values proposed by participating nodes have to intersect in at least one correct node to enable the system to make a decision) [24, 129, 130]. These implementations result in a quadratic communication complexity [24, 99], while optimized versions were barely able to reach a linear complexity [25].

Scalable Byzantine Reliable Broadcast (sBRB) [85] extends the classical BRB primitive to address the scalability limitations inherent in traditional implementations. sBRB protocols maintain the core safety properties of BRB while reducing per-node communication complexity from linear $O(n)$ to logarithmic $O(\log n)$. The key innovation in this approach is the use of randomization across nodes instead of requiring agreement across majorities.

Scalable Byzantine Reliable Broadcast Overview

Scalable Byzantine Reliable Broadcast (sBRB) [85] is a broadcast primitive that falls in the category of randomized algorithms. sBRB aims to transmit a message over a network of nodes, such that the same version of the message is delivered by all correct nodes.

By making use of sBRB abstraction, we achieve consensus at logarithmic communication cost. Being a probabilistic approach helps bypassing the impossibility result of agreement in the presence of one faulty process [69] by guaranteeing probabilistic properties rather than deterministic ones. Moreover, the peer sampling method used by sBRB to sample communicating peers enables each node to pick a sample of peers (following the Poisson distribution) to collect feedback for message delivery, instead of expecting confirmation from a majority quorum.

Unlike traditional consensus protocols that require resource commitment, sBRB achieves probabilistic agreement through multiple phases of communication with significantly lower overhead. The protocol works by having each node communicate with small randomly sampled subsets of nodes rather than the entire network.

The protocol relies on the statistical properties of random sampling, allowing correct information to propagate through the network over multiple rounds. More specifically, using multiple rounds of communication with different random samples significantly increases the likelihood that correct information propagates throughout the network [80].

This sampling approach is considered a key feature for a scalable design, as it allows reduced messages exchange to validate a transaction. The samples are significantly smaller in size compared to quorums. The protocol operates on the principle that a random sample of sufficient size will, with high probability, reflect the state of the entire network. The importance of this feature becomes increasingly important as the network grows and the gap between quorum and sample sizes widens. That also results in a low latency due to the minimal messages exchange. Indeed, a node does not have to wait for the responses of the majority to deliver or discard a message. Rather, it communicates with a few sampled peers that reflect the overall network state.

Also, unlike in PoW and PoS systems where validators are privileged based on their dedicated resources (E.g., computation power, stake), adopting sBRB enables all nodes to be potential validators. Nodes

collect confirmation from their communication samples to validate a transaction, giving equal chances to all users to participate in the agreement process. sBRB does not impose a total order among all transactions in the system. This design choice is intentional. For asset-transfer applications, correctness relies on preventing conflicting spends and ensuring transaction agreement, rather than maintaining a globally ordered ledger [86]. Consequently, LighTx avoids the overhead of consensus-based ordering while preserving the guarantees required for secure asset transfer.

sBRB represents an appealing alternative to consensus-based approaches for the asset-transfer problem. Rather than establishing a globally ordered ledger, sBRB guarantees reliable dissemination and agreement on individual transactions while incurring only logarithmic communication overhead. Since sBRB relies on probabilistic sampling, its guarantees hold with high probability and are therefore expressed using an error parameter ϵ.

More specifically, sBRB provides three fundamental properties: (1) ϵ-*Validity*: If a correct sender broadcasts a transaction tx, then all correct processes deliver tx with probability at least $1-\epsilon$. (2) ϵ-*Consistency*: With probability at least $1-\epsilon$, no two correct processes deliver conflicting versions of the same transaction. That is, if a correct process delivers tx, every correct process delivering the corresponding broadcast instance delivers the same transaction. And (3) ϵ-*Totality*: If a correct process delivers a transaction tx, then all correct processes eventually deliver tx with probability at least $1-\epsilon$.

Totality means that if one correct node delivers a message, all correct nodes eventually deliver that same message. It does not mean that all correct nodes deliver all transactions in the same global order.

The use of sBRB in LighTx follows the observation that asset transfer is a weaker abstraction than state-machine replication. A blockchain protocol such as Ethereum uses consensus to establish a total order of blocks because smart-contract transactions may interact through arbitrary shared state. By contrast, in a pure asset-transfer object, the main safety violation is double-spending: two incompatible outgoing transfers from the same account or asset should not both be accepted. Therefore, the system does not necessarily need to order all transactions globally; it only needs to ensure consistency among dependent or conflicting transfers. This distinction is formalized by Guerraoui et al. in AT2 [83], where decentralized asset transfer is shown to be implementable without consensus, using broadcast abstractions instead of total-order consensus. AT2 explicitly argues that consensus is not needed to implement decentralized asset transfer and shows that an asset-transfer object has consensus number one; in Byzantine message-passing, it builds asset transfer using broadcast primitives instead of total-order consensus. A related result in [86] also shows that, when only the account owner can withdraw from an account, consensus is not necessary to prevent double-spending.

3.3.3 Broadcast Communication Phases

sBRB protocol operates through a multi-phase process: (1) the *Dissemination phase* for message propagation, (2) the *Confirmation phase* that ensures agreement on the same message, and (3) the *Commitment and finalization phase* that guarantees total message delivery across all correct nodes. The combination of these phases establishes robust Byzantine fault-tolerant broadcast semantics with logarithmic complexity.

Dissemination Phase. initiates the broadcast operation through a gossip-based algorithm that distributes a message m (initiated by the sender) across the network via a dynamically generated Erdös-Rényi random graph [66]. The aim of this phase is to create an efficient probabilistic dissemination pattern that ensures rapid message propagation through a logarithmic-size random samples.

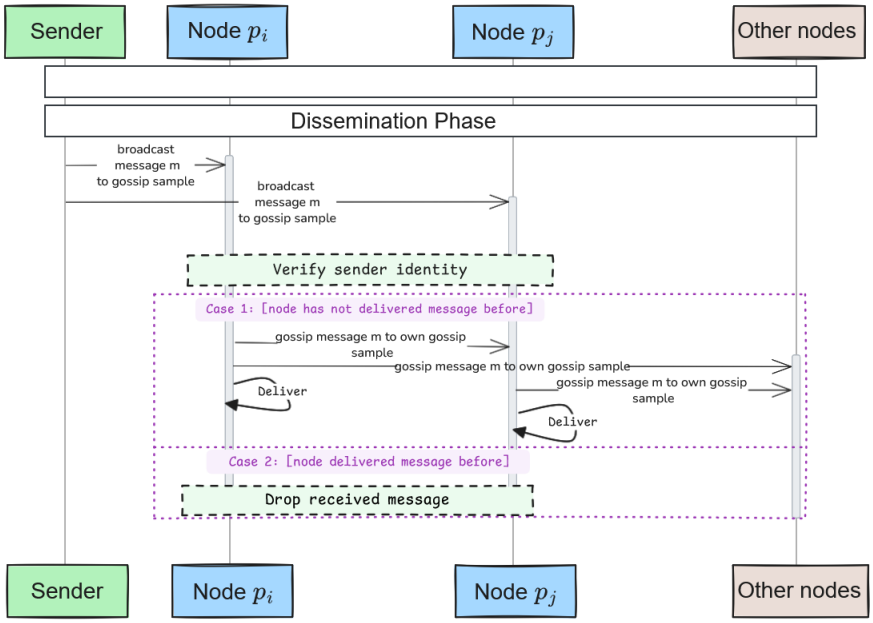


Figure 3.1: Dissemination Phase of sBRB. The sender broadcasts message m to nodes in its gossip sample (e.g., p_j). Receiving nodes verify the sender's identity. If valid and message is new (Case 1), then deliver m, and gossip m to their own gossip sample. If invalid or duplicate (Case 2) then drop m. Each recipient repeats the same verification/delivery logic. Messages propagate through gossip samples in iterative waves. Propagation naturally ceases when all nodes have either delivered or dropped the message.

Each node independently selects a random subset of peers (gossip sample) of size $g = O(\log n)$, where n is the total number of nodes. This sample size balances redundancy and scalability; small enough to avoid $O(n^2)$overhead but large enough to ensure probabilistic coverage.

The broadcast begins when the sender signs message m and transmits it to all members in its gossip sample. When a node receives a gossip message, it first verifies the sender's identity by validating the digital signature. Then, it accepts and delivers the message locally if it has not previously received it. Lastly, the node propagates the message only once to all members of its own gossip sample. To maintain efficiency, upon first receipt, the protocol forwards and delivers only the first received copy of each message and discards subsequent duplicates caused by overlapping samples.

Confirmation Phase. Following the initial dissemination, this phase ensures consistency across the network by guaranteeing with high probability that each correct node delivers the same message, if it delivers one at all, even in the presence of a Byzantine sender.

Upon initialization, each node independently selects an *echo sample* of peers, sized at $e = O(\log n)$, using a cryptographically secure randomness source Ω. When a node validates the receipt of a message m (from the dissemination phase), it generates a signed $echo(m)$ and transmits it to all nodes in its echo sample. Correct nodes echo only a single message m in a communication round.

Concurrently, each node collects incoming echoes from peers that included it in their echo samples. A node delivers message m only after collecting a threshold number of matching echoes for the same message m. By requiring independent confirmation from multiple randomly selected nodes before delivery, the confirmation phase ensures with high probability that if any correct node delivers a message, it must be

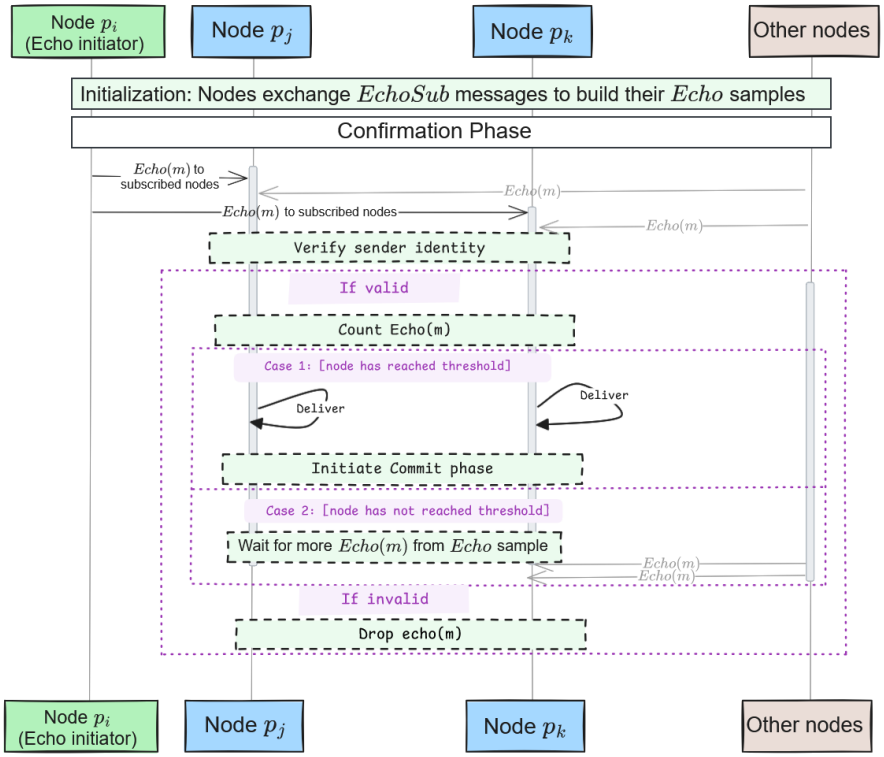


Figure 3.2: Confirmation Phase of sBRB. After delivering the gossip message, nodes initiate the confirmation phase by sending *echo*(m) messages to nodes in their Echo subscription samples. Upon receiving *echo*(m), nodes verify the sender's identity. If valid, nodes count the accumulated *echo*(m) messages. If the count reaches the threshold θ_{echo} and the node has not already echoed (Case 1), the node delivers the message and initiates the *Commit phase*. If the threshold has not been reached (Case 2), the node waits for additional *echo*(m) messages from members of its Echo sample. Invalid *echo*(m) messages are dropped.

the message that most correct nodes have received and delivered. Also, Byzantine nodes cannot cause the correct nodes to deliver different messages. Finally, the confirmation process maintains a logarithmic communication complexity per node.

Commitment and Finalization Phase. The commitment phase represents the final stage of the sBRB protocol. This phase guarantees that if any correct node delivers a message, then every correct node eventually delivers that message with high probability.

Upon initialization, each node establishes two distinct random samples of logarithmic size: A ready sample, for collecting readiness indicators, and a *delivery sample* for confirming final message delivery.

Upon validating m in the Confirmation Phase, a node signs a *ready*(m) message and broadcasts it to its ready sample. Nodes collect incoming *ready*(m) signals. Upon collecting a threshold number of ready messages for the same message m, a node transitions to a *ready* state for that message and re-broadcasts *ready*(m) to its own ready sample. A node delivers message m irreversibly only after receiving sufficient ready messages from its delivery sample, confirming global consensus.

The cascading ready message propagation creates a powerful feedback mechanism that overrides potential attempts by Byzantine nodes to prevent message delivery. Once sufficient correct nodes become ready for

a message, the ready state becomes ”*contagious*” and inevitably spreads throughout the network, ensuring totality despite Byzantine interference.

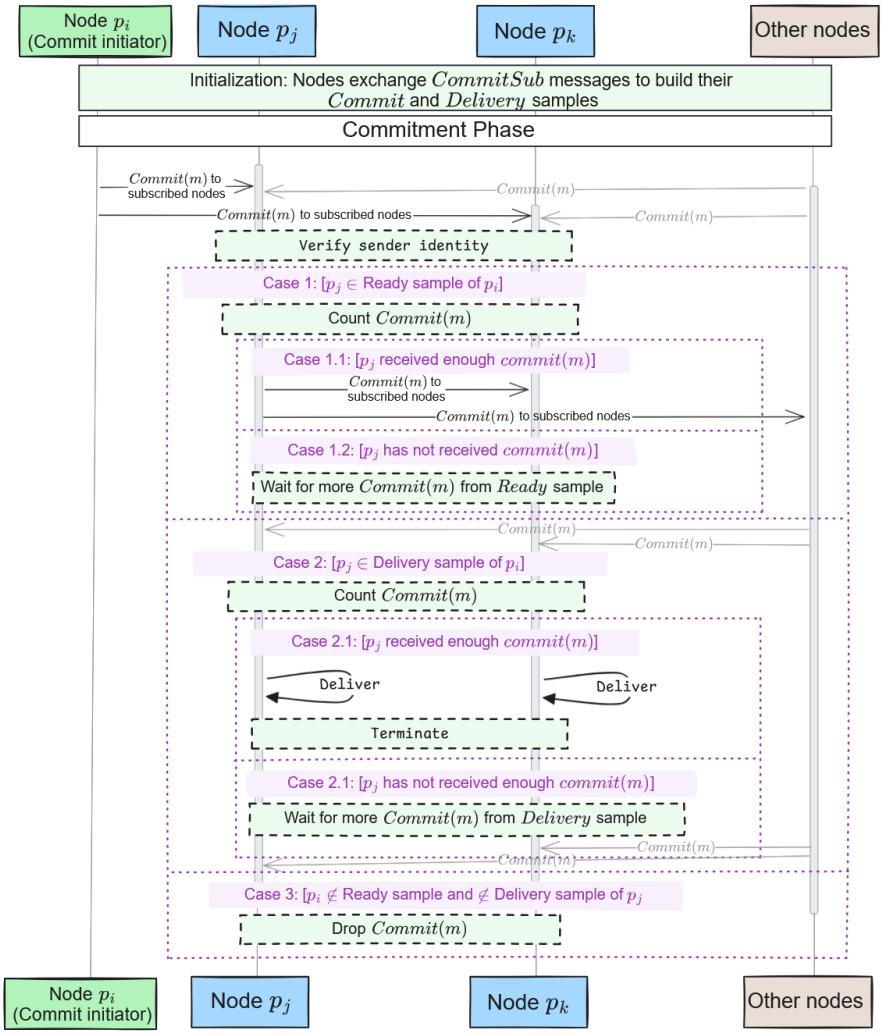


Figure 3.3: Commitment and finalization phase in sBRB. After delivering the Echo message *echo*(m), nodes initiate the commitment phase by sending *commit*(m) messages to their Commit subscription sets. Upon receiving *commit*(m), nodes verify the sender's identity and apply conditional logic based on sample membership. Case 1 applies when the sender belongs to the receiver's Ready sample: if enough *commit*(m) messages are received (Case 1.1), the node sends *commit*(m) to its own subscription set; otherwise it waits for more *commit*(m) messages (Case 1.2). Case 2 applies when the sender belongs to the receiver's Delivery sample: if enough *commit*(m) messages are received (Case 2.1), the node delivers the message and terminates the protocol; otherwise it waits for additional *commit*(m) messages (Case 2.2). Case 3 handles senders outside both samples by dropping the *commit*(m) message.

3.3.4 Technical Limitations of the Broadcast Primitive

As we described the broadcast primitive, we identified two main constraints that hinder its applicability in practical, large-scale systems.

In sBRB, every submitted transaction is carried by a single broadcast instance. This helps to differentiate between legitimate send operations and inconsistent message dissemination (in the case of a Byzantine sender). While this isolation simplifies safety guarantees, it poses significant challenges in environments where nodes issue concurrent transactions.

In concurrent environments, nodes issue multiple transactions simultaneously. Each transaction triggers an independent broadcast instance, resulting in a flood of interleaved sBRB messages (e.g., gossip, echoes, ready signals). Recipients face ambiguity, as messages lack inherent metadata to correlate them with their originating transactions.

On the other hand, sBRB requires additional mechanisms to prove robustness in high churn networks. sBRB tends to be more suitable for private networks [85] and claims that in an open environment with slow churn, the issue could be approached in the same fashion as in private networks to deal with the effect of openness. Nevertheless, in open environments, robustness challenges get even harder, through the exposure to dynamic malicious users. Open environments expose the protocol to dynamic adversarial behavior, particularly Sybil attacks, where malicious entities can create multiple identities to spread and validate inconsistent transactions. This requires proper inspection of the real identity of nodes in order to preserve system robustness in networks with churn.

3.4 The LighTx Architecture

LighTx represents a novel approach to transaction processing in distributed environments, designed to overcome the limitations of traditional broadcast primitives while maintaining security and performance in open networks. By combining sBRB primitive with innovative routing and reputation mechanisms, LighTx delivers a lightweight yet robust solution for transaction transfer across distributed systems.

LighTx leverages the security guarantees of the sBRB primitive while addressing its key limitations through three complementary components:

- Enhanced Routing Layer: LighTx incorporates a streamlined routing mechanism that enables concurrent transaction submission, solving the transaction isolation problem inherent in standard sBRB implementations.
- Proof of Bandwidth: To counteract Sybil attacks in open networks, LighTx implements a Proof-of-Bandwidth (PoB) system that monitors communication patterns between nodes. The Proof-of-Bandwidth (PoB) works by measuring bandwidth fluctuations, the system can identify anomalous behavior that suggests a node is interacting with an unusually varying number of peers, which may indicate a Sybil attack.
- Reputation System: Building on bandwidth measurements, LighTx establishes a comprehensive reputation framework based on EigenTrust [103]. Trust scores are periodically computed and distributed throughout the network, creating a self-regulating mechanism that rewards honest participation and penalizes malicious behavior.

3.4.1 Concurrent Broadcast Channels

The design of LighTx draws primarily on sBRB abstraction [85]. We customize sBRB via setting up a routing mechanism for messages identification management to support concurrency in transactions

Algorithm 1 LIGHTX ROUTING

```
1: Initiate sBRB broadcast:
2:     create broadcast instance sbrb with broadcast channel ID ⟨sbrb_id⟩
3:     routingTable[sbrb_id] ← sbrb
4:     sbrb.broadcast ⟨tx|sbrb_id⟩                                  ▷ The sender

5: Upon receipt of sBRB message with sbrb_id                        ▷ All nodes
6:     sbrb ← ROUTE(sbrb_id)
7:     proceed sBRB broadcast process with broadcast instance sbrb

8: function ROUTE(sbrb_id)
9:     if routingTable contains sbrb_id then
10:        return getValue(sbrb_id)
11:    else
12:        routing Table.add(sbrb_id : new sbrb)
13:        return getValue(sbrb_id)
14:    end if
15: end function
```

submission. The routing layer is an instance-separation mechanism. As LIGHTX enables concurrent transaction broadcasts, messages must be demultiplexed according to their broadcast identifier. The mapping ensures that the issued sBRB messages (gossip, echo and commit messages) for one transaction cannot be processed in the state machine of another transaction. The routing layer therefore preserves the integrity of each sBRB instance under concurrency. Algorithm 1 describes the routing mechanism in LIGHTX.

The sender s initiates a broadcast operation by creating a broadcast instance *sbrb* identified with a unique broadcast channel identifier $sbrb_{id}$, then adds the new pair $\{sbrb_{id} : \textbf{\textit{sbrb}}\}$ to the routing table (line 3). Next, the sender broadcasts the transaction labelled with its broadcast channel ID using the sBRB primitive as described in [85] (line 4). The broadcast channel ID is generated by combining the broadcast instance hash code with the current timestamp. When a broadcast operation is triggered, the broadcast instance is created, by the sender s, associated with its broadcast channel ID. The broadcast channel ID is unique across all nodes.

Upon receipt of any sBRB message, the receiver consults its routing table. If the routing table contains the broadcast channel ID of the received sBRB message, the corresponding broadcast instance handles the sBRB message (line 10). Otherwise, a new broadcast instance will be added to the routing table to handle messages originating from this broadcast channel (line 12). Once the broadcast channels are identified, the message processing continues following the *sBRB* primitive (line 7).

Routing invariant. For any two broadcast identifiers $sbrb_{id1} \neq sbrb_{id2}$, a correct node processes messages tagged with $sbrb_{id1}$ only in the sBRB instance associated with $sbrb_{id1}$, and messages tagged with $sbrb_{id2}$ only in the sBRB instance associated with $sbrb_{id2}$. Therefore, concurrent transactions do not interfere at the protocol-state level. This invariant enables LIGHTX to execute many independent transaction transfers in parallel.

3.4.2 Proof-of-Bandwidth Scheme

To resolve the issues posed by Sybil attack while maintaining the major feature sBRB offers which is scalability, we propose a lightweight PoB-based reputation system to prevent malicious peers from performing a Sybil attack and thus, manipulating transactions validation. Unlike traditional Sybil-resistant mechanisms such as Proof-of-Work [143] or Proof-of-Stake [33], nodes are not exhausted by investing a huge amount of energy and CPU power or financial barriers, to prove their identities. Our system leverages bandwidth utilization patterns to distinguish legitimate participants from malicious Sybil identities. The concept drives from the intuition that transmitted transactions should be proportional to the used bandwidth resources. A genuine identity should reflect stable bandwidth measurement records, while a node running multiple fake identities would have fluctuating bandwidth measurements caused by flow interference, high load and medium occupancy.

Bandwidth Measurements. As a first step, we set up a bandwidth evaluation method, to measure the available bandwidth of peers. Over multiple rounds, we perform bandwidth measurements and track bandwidth changes of the evaluated nodes. Multiple bandwidth estimation techniques exist in the literature depending on application usage. Most of the bandwidth estimation techniques are based on probe messages. We adopt the one-way-delay method [108] as it represents a low overhead compared to other existing methods [104, 154].

Receivers may misbehave by faking timestamps of the probe messages. However, given that the adversary does not know about the measured Round Trip Time (RTT) on the sender side. Also, given that probe messages are sent on various throughput speed, then inducing random timestamps to the probe reply comes against the peer itself. Therefore, measuring a stable bandwidth capacity would be preferred.

Local Scores Assignments. Bandwidth measurements are conducted over r consecutive rounds. For each peer p_j, a node maintains a bandwidth vector $BW_j = [bw_1, bw_2, \ldots, bw_r]$, where bw_k represents the measured bandwidth of p_j in round k. When the measurement phase is finished, each assessed node will be assigned an initial local score $\omega_{ij}^{(0)}$. We set $\omega_{ij}^{(0)}$ to be equal to the mean μ_{bw_0} over all the assessed peers in the first measurement round. Then, scores are updated based on the variance of the measurements taken over r rounds and the maximum assessed bandwidth of each peer. The updated local scores ω_{ij} serve as inputs to the decentralized reputation system, where they are aggregated into global reputation scores. We translate the score update by Expression 3.1.

$$\omega_{ij} = \begin{cases} \omega_{ij}^{(0)} - \sigma_{bw_j} BW_{j_{\max}}, & \text{if } \sigma_{bw_j} > \alpha, \\ \omega_{ij}^{(0)}, & \text{otherwise.} \end{cases} \tag{3.1}$$

Where ω_{ij} is the updated local score, $\omega_{ij}^{(0)}$ is the initial local score, σ_{bw_j} is the variance of bandwidth measurements measured over r rounds. σ_{bw_j} quantifies the stability of node p_j's bandwidth. $bw_{j_{max}}$ is the maximum bandwidth capacity measured of node p_j and α is fluctuations tolerance factor which indicates the permissible variance threshold allowed without penalty.

If the variance of the measurements σ_{bw} is negligible, the initial scores are maintained. When the fluctuations are considerable, the new score takes into account the variance of bandwidth measurement σ_{bw} and the maximum measured bandwidth bw_{max}. That implies that nodes with high bandwidth are heavily penalized when they show some anomalies, since they possess greater potential to disrupt the network if malicious. Hence, nodes exceeding α are penalized proportionally to both their instability (σ_{bw}) and disruptive potential (bw_{max}).

The intuition of this methodology assumes that a genuine node operates within stable bandwidth limits, consistently dedicating a predictable fraction to protocol tasks. While nodes running multiple Sybil identities split bandwidth, causing detectable fluctuations. Malicious nodes operating multiple identities exhibit elevated σ_{bw} due to bandwidth fragmentation or toggling between attacking and idling. For example, a node splitting 100 Mbps across 10 Sybils would show variance spikes as it reallocates capacity. Algorithm 2 describes the *Proof-of-Bandwidth* scheme and the local scores update during assessment phase.

Algorithm 2 PROOF-OF-BANDWIDTH

Parameters. α : Fluctuation tolerance factor
R : Number of rounds

procedure PROBE()
  for each node p_i **do**
    for r rounds **do**
      Send *Probe Message* to nodes of ***sbrb*** samples
    end for
  end for
end procedure

function SCOREUPDATE(σ_{bw}, bw_{max})
  if $\sigma_{bw} > \alpha$ **then**
    return $\omega_{ij}^{(0)} - \sigma_{bw}.bw_{max}$
  else
    return $\omega_{ij}^{(0)}$
  end if
end function

Upon receipt of a *Probe Message*
if $bw < R$ **then**
  $bw.add(bw_i)$
else
  $bw_{max} = max(bw)$
  Calculate variance σ_{bw} of bw
  $\omega_{ij}^{(0)} = \mu_{bw_0}$
  ω_{ij} = SCOREUPDATE($\sigma_{bw}, bw_{max}, \alpha$)
end if

3.4.3 Reputation System

To support the *Proof-of-Bandwidth* mechanism LIGHTX integrates a reputation system to evaluate aggregated local bandwidth-based scores into a global network view. Our approach is inspired by EigenTrust [103] that builds upon the notion of transitive trust. Nodes consider the opinion of remote nodes they trust, by collecting their local reputation scores. Based on the remote evaluation, each node aggregates the reputation score from its own local scores and the scores received from remote nodes, then shares it back with nodes that trust him. By iteratively repeating this process for a certain number of rounds, it converges

to the *eigenvector* of the score matrix. Each node in the system converges to a global reputation score, based on which they can make a decision on the malicious nodes.

Score Aggregation. To prevent adversaries from poisoning the network by assigning arbitrary high or low scores, we first normalize the local reputation scores in a range between zero and one. We define m_{ij} as the normalized score assigned by node p_i to node p_j in Equation 3.2. We note that score normalization also discards all negative scores (set them to zero). Also, in the edge case where $\sum_{k=1}^{n} \max(\omega_{ik}, 0) > 0$ (all scores negative), default to uniform distribution $m_{ij} = \frac{1}{n}$, preventing division-by-zero errors.

$$m_{ij} = \frac{max(\omega_{ij}, 0)}{\sum_{k=1}^{n} \max(\omega_{ik}, 0)} \tag{3.2}$$

To aggregate scores, each node requests the peers it evaluated for their local scores, weights their opinions and take them into consideration to compute the global score value by iterating over the matrix of normalized scores M.

To compute a network-wide reputation consensus, each node iteratively aggregates local trust scores from peers using the normalized score matrix M. This matrix encodes pairwise trust relationships, where each entry m_{ij} represents node p_i's normalized trust in node p_j.

The process begins with an initial uniform score vector $m^{(0)}$, which reflects no prior bias about node trustworthiness. In the first iteration, we compute the score vector $m^{(1)}$ by multiplying the local score matrix M with the remote score vector m as $m^{(1)} = M.m$. Then, for each iteration, p_i updates its global score vector $m^{(2)} = M \cdot m^{(1)} = M^{(2)} \cdot m$ to $m^{(n)} = M^{(n)} \cdot m$. The process is repeated until the score vector stabilizes, when the difference between successive vectors falls below a threshold ϵ as shown in Expression 3.3.

$$\left\| m^{(k+1)} - m^{(k)} \right\| < \varepsilon \tag{3.3}$$

At this stage, the network converges to one global vector which corresponds to the principal *eigenvector* of matrix M.The eigenvector $m^{(*)}$ reflects the global reputation where each node's score is proportional to the collective trust placed in it by the network.

Byzantine Resilient Distributed Score Aggregation. To avoid the impact of *sink nodes* (i.e nodes who share very low scores about other nodes, influencing their global scores), we use predefined set of pre-trusted nodes P (e.g., bootstrap servers or long-standing participants). These nodes are assigned a static, uniform trust distribution via vector $\overrightarrow{\rho}$. We define ρ_i the elements of vector $\overrightarrow{\rho}$ a follows:

$$\rho_i = \begin{cases} \frac{1}{|P|} & : \quad \text{if} \quad i \in P \\ 0 & : \quad \text{otherwise} \end{cases} \tag{3.4}$$

The use of vector $\overrightarrow{\rho}$ as the start vector for the score aggregation stabilizes quickly and leads to faster convergence [103, 26]. Additionally, LIGHTX introduces a damping factor $d \in [0, 1]$ [26] as described by Equation 3.5, to counter coordinated attacks such as Sybil collectives inflating mutual scores. The d term scales down scores derived from peer evaluations, limiting the impact of manipulated votes. While $1 - d$

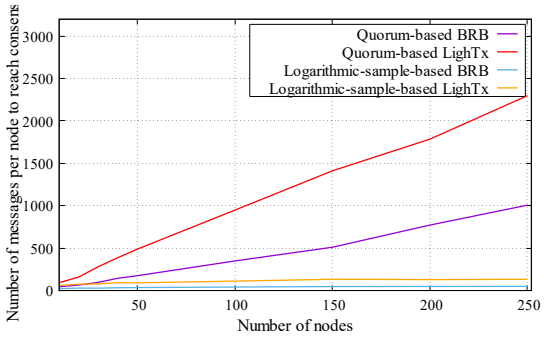


Figure 3.4: Number of exchanged messages per node in sample-based vs. quorum-based LIGHTX.

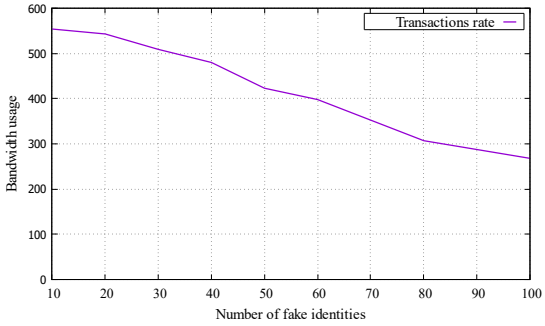


Figure 3.5: Transaction rate as a function of the network size.

injects fixed trust into the pre-trusted nodes, preserving their stabilizing role.

$$M^{(k+1)} = d \cdot Mm^{(k)} + (1-d)\overrightarrow{\rho} \tag{3.5}$$

We describe the distributed version of the score aggregation in Algorithm 3.

Algorithm 3 REPUTATION SYSTEM

Parameters A_i : Nodes evaluated by node p_i
B_i : Nodes who evaluated node p_i
ϵ : Convergence precision
d : Damping factor

for Each node p_i **do**
  Query all nodes $p_j \in A_i$ for m_{ji} and $m_j^{(0)} = \rho_j$
  repeat
    $m_i^{(k+1)} = d(m_{1i}m_1^{(k)} + m_{2i}m_2^{(k)} + .. + m_{ni}m_n^{(k)}) + (1-d)\rho_i$
    send $m_{ij}m_i^{(k+1)}$ to all nodes $p_j \in B_i$
    wait for all nodes $p_j \in A_i$ to return their $m_{ji}m_j^{(k+1)}$
  until $|m_i^{(k+1)} - m_i^{(k)}| < \epsilon$
end for

3.5 Evaluation

In this section, we present a comprehensive evaluation of LIGHTX. We implemented LIGHTX using native Java, creating a fully functional prototype to assess its performance characteristics under various conditions. The source code for our implementation is publicly available at https://github.com/ImaneElabid/LighTx. The PoB mechanism and reputation system components were rigorously evaluated through detailed simulations across a range of representative data scenarios. we assess the performance of our system. Our prototype was implemented in Java language.

Experimental Setup. All experiments are conducted on a computer running Windows 10 OS, with

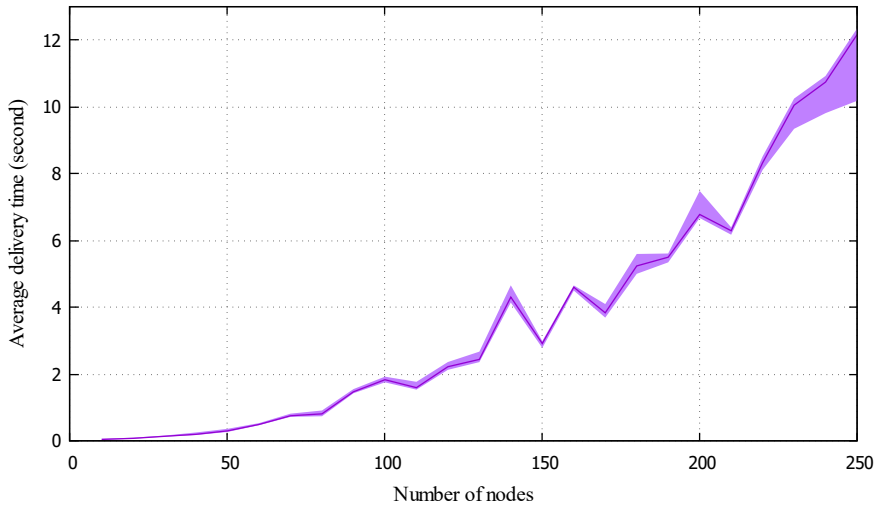


Figure 3.6: Average latency as a function of the network size.

Intel(R) Xeon(R) W-2123 CPU 3.60GHz and 32.0G of RAM. For the sake of simplicity, we withdraw link authentication property to dispose of authentication certificate setups. We use therefore, TCP as transport protocol. In what follows, we detail the circumstances of the experiment and analyze the results. In the correct execution model, a sender s broadcasts a transaction and distributes it among the network. We assume that all nodes of the network behave correctly.

Communication Complexity. All samples sizes are set to be equal and logarithmic in size with respect to the number of nodes. With this setting, we vary the number of nodes from 10 to 250 nodes and we measure the number of received messages per process for each broadcast instance. Fig. 3.4 depicts that contrarily to quorum-based sBRB, the number of messages of LightTx using sample-based sBRB does not increase with the total number of nodes. This is because each node communicates only with members of its samples (of logarithmic size). Even with such a minimal message exchange, LightTx achieves agreement.

Latency. For the system latency, we measure the time elapsed between the broadcast and the delivery operation. We plot the delivery time averaged over multiple runs. Fig. 5.6 shows that latency increases with the number of nodes. For example, we see that it takes on average 12 seconds to deliver a consistent transaction over a network of 250 nodes. Note that part of the latency augmentation is generally due to external causes, such as concurrency and multi-threading issues related to hardware limitation. In our simulation environment, as the system grows bigger, concurrency issues become glaring. We presume that these external delays can be considerably reduced in a distributed architecture.

Data Rate. The obtained transaction rate is shown in Fig. 3.5 where we note that the application reaches a throughput of more than 500 tps when the number of nodes is relatively small. Transaction rate decreases with the number of nodes, but remains clearly higher compared to Bitcoin which can only handle around 7 tps [49] or Ethereum that reaches 20 tps [194].

This performance gap is explained by a fundamental difference in ordering guarantees. For instance, Ethereum maintains a globally ordered ledger through consensus and replicated state-machine execution, which serializes every operation and incurs multi-round coordination overhead that directly limits its peak throughput. By contrast, LightTx targets the narrower but practically important class of asset-transfer

applications. For this class, the central correctness condition is not global serialization, but rather the prevention of double-spending. Because independent transfers on disjoint accounts commute, LIGHTX can safely relax the ordering requirement without violating safety. Instead of imposing a total order over all transactions, it leverages sBRB to agree only on conflicting or dependent transfers, avoiding unnecessary coordination for the vast majority of non-conflicting operations. Consequently, the throughput values we report do validate the theoretical insight that eliminating consensus-based ordering, directly translates into measurable gains in sustained delivery rate and scalability.

Adversarial Execution. In a centralized model of our reputation system, the network converges to global trusts within few rounds (about 5 rounds, see Fig. 3.7) the decentralized version of the algorithm.

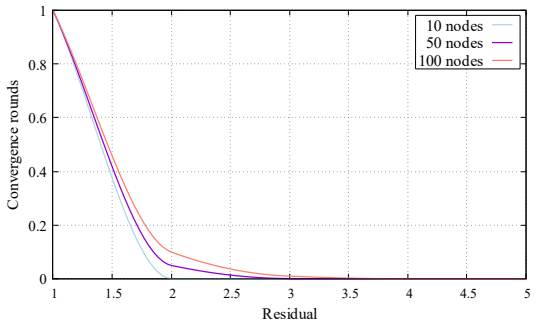


Figure 3.7: Convergence rounds in the centralized version of the reputation system.

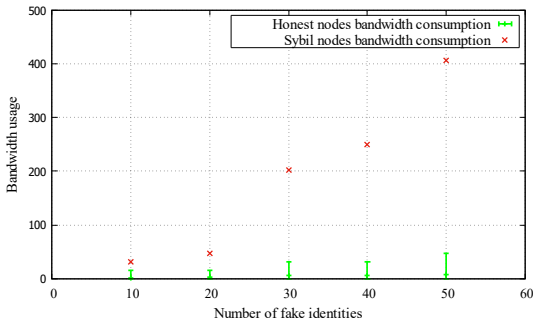


Figure 3.8: Correct vs. Sybil nodes bandwidth consumption for transaction transfer.

In this simulation, in a network of 100 nodes having equal bandwidth resources, we assess bandwidth usage to validate one transaction. We record the bandwidth measurements of all nodes on the network, while varying k_{id} the number of identities created by a Sybil attacker. Fig. 3.8 shows the difference between honest nodes bandwidth records and a Sybil node records as we increase the fake identities k_{id} of a Sybil attacker. We note that as k_{id} grows, the ability to detect Sybil nodes increases, as they become overloaded with the different flows they receive and that lowers their transmission rate. Concretely, a Sybil node p_k should run a large number of fake identities k_{id} to take over the network. p_k in this case, communicates simultaneously with $S_{avg}.k_{id}$ (where S_{avg} is the average size of communication samples).

3.6 Conclusion

In this chapter, we presented LIGHTX, a Byzantine-resilient transaction transfer system designed to achieve consensus with minimal resource overhead. It departs from traditional consensus paradigms that require high energy consumption, large quorums, or complex leader elections by establishing trust through measurable and fair participation. The architecture of LIGHTX enables each component to be reused or integrated into other lightweight decentralized frameworks. LIGHTX demonstrates that scalable asset transfer can be achieved without maintaining a globally ordered ledger, supporting the growing body of work showing that consensus is not a fundamental requirement for secure asset-transfer systems. Its broadcast primitive achieves Byzantine-resilient message delivery through randomized peer sampling with logarithmic message complexity per node. The routing layer enables concurrent transaction processing using unique channel identifiers, removing ambiguity in multi-transaction environments.

Sybil resistance is achieved through bandwidth stability analysis, where trust derives from consistent

communication patterns rather than their bandwidth capacity. A decentralized reputation system ensures global trust convergence using eigenvector-based propagation of these bandwidth metrics, reducing the impact of collusive behavior. This allows devices with varying capabilities, from low-power nodes to high-bandwidth servers, to participate meaningfully based on stable contribution rather than absolute capacity. LightX's tunable parameters, such as sampling rate, reputation decay, and bandwidth thresholds, allow adaptation to application-specific priorities including low-latency finality for real-time IoT and stronger resilience for financial systems. This flexibility avoids a fixed trade-off, balancing throughput, security, and efficiency as needed.

Performance evaluation shows that LightX achieves over 500 tx/s with latency under a few seconds, while maintaining robustness against Byzantine adversaries. A key limitation is that the protocol processes transactions individually rather than batching them into blocks. While this approach simplifies consensus analysis and aligns with micropayment and IoT use cases, it constrains throughput compared to block-based architectures. The current design also assumes nodes behave honestly without formal incentives. Although reputation encourages cooperative behavior, the lack of economic motivation may limit long-term sustainability in open environments.

The subsequent chapter addresses these limitations by introducing transaction batching to improve throughput while preserving security. The system is further extended to support full participation by heterogeneous, resource-constrained devices, with incentive mechanisms explored to sustain their engagement.

4 Mobile-Centric Blockchain: From Clients to Full Nodes

The rapid proliferation of smartphones and low-power devices has produced a dense and globally distributed computational infrastructure significant aggregate processing resources. Modern mobile platforms benefit from advances in battery technology [125], energy-efficient hardware [90, 89], and optimized software stacks [94, 17, 145], making them increasingly capable of supporting decentralized applications. Yet, mainstream blockchain systems remain largely incompatible with mobile environments. Consensus protocols impose high computational, communication, and storage overheads, limiting active participation to resource-rich nodes and reinforcing structural hierarchies that exclude constrained devices from full validation roles [202, 165, 11]. This exclusion undermines the decentralization and fairness principles blockchain aims to preserve.

The fundamental question driving this research is: *Can resource-constrained devices participate as full nodes in blockchain consensus without compromising security or performance?* To this end, we introduce PocketChain, a resource-efficient blockchain architecture that enables mobile devices to participate as active nodes in consensus without specialized hardware, persistent connectivity, or elevated resource budgets. PocketChain builds on a two-phase consensus protocol $\mathscr{P}ccp$, combining an endorsement phase for concurrent block proposal and conflict filtering, with a broadcast phase using a scalable reliable broadcast [85], extended with parallel virtual broadcast channels, block-level batching, and incentive-driven participation. This structure allows nodes to achieve consensus efficiently, while maintaining robustness against Byzantine behavior. PocketChain eliminates rigid node hierarchies through dynamic, role-based participation where contributions scale with available resources, allowing mobile devices to sustain participation with modest CPU and energy footprints.

While mobile devices represent our primary use case, PocketChain accommodates the full computing spectrum from constrained devices to high-performance servers. Figure 4.1 illustrates PocketChain running across a heterogeneous network.

Table 4.1 presents a preliminary analytical comparison of PocketChain with traditional blockchain systems, emphasizing its role separation, mobile compatibility, and scalable parallelism. These features provide the foundation and motivation for PocketChain architecture presented in the following sections, where comprehensive analytical and empirical validation demonstrates their practical effectiveness. This chapter details the system model, $\mathscr{P}ccp$ consensus, incentive structure, and resource optimizations, and concludes with a performance evaluation demonstrating PocketChain's scalability and efficiency.

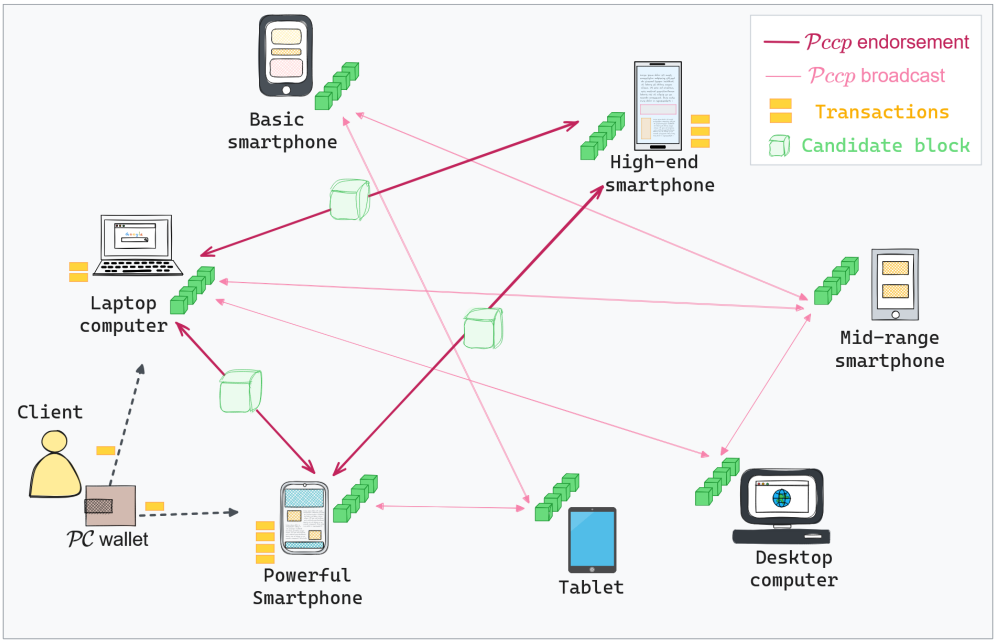


Figure 4.1: PocketChain operating across a heterogeneous peer-to-peer network. The network comprises various resource-constrained and resource-rich devices, including smartphones, tablets, laptops, and desktop computers.

4.1 Related Work

The integration of blockchain with mobile devices introduces critical challenges due to constrained computational resources, limited battery life, and intermittent connectivity. Recent research efforts attempt to address these constraints through lightweight consensus, scalable architectures, and mobile-centric frameworks that balance performance, security, and decentralization.

Table 4.1: Preliminary comparison between traditional blockchains and PocketChain.

Feature	Traditional Blockchains [143, 30]	PocketChain
Consensus roles	Uniform: all nodes mine/validate	Decoupled: Announcers propose, Processors validate
Resource demands	High: ASICs (PoW) / +32ETH (PoS)	Low: runs on constrained devices
Throughput	7–30 TPS (serial execution)	+2000 TPS (parallel VBChs)
Finality time	Minutes (PoW) / Seconds (PoS)	1–2 seconds (endorsement + sBRB)
Energy per tx	707kWh (Bitcoin) / 0.02kWh (Ethereum)	0.027mWh (mobile-optimized)
Conflict resolution	Forks (orphan chains)	Pre-consensus (endorsement)
Mobile support	None	Native participation (Processor role)

4.1.1 Lightweight Consensus

Systems like IOTA [153] use DAG-based structures to facilitate parallel validation and achieve high throughput without miners. Despite its low fees and asynchronous model, IOTA faces challenges with coordinator centralization, transaction finality, and tip selection complexity. In contrast, PocketChain achieves scalability via parallel virtual broadcast channels, combining explicit pre-consensus endorsement with scalable broadcast to improve fairness, inclusion, and energy efficiency on mobile devices. Recent works promoting lightweight asset transfer solutions build directly on Scalable Byzantine Reliable Broadcast (sBRB). AT2 [83] proposed to eliminate heavy consensus by using sBRB to ensure consistent transaction delivery in asynchronous networks, but lacks conflict resolution, limiting its applicability to concurrent workloads. LightX [64], on the other hand, couples sBRB with proof-of-bandwidth-based reputation system to encourage honest message relaying in lightweight environments, but incurs per-transaction validation overhead. PocketChain uniquely integrates sBRB into a two-phase consensus protocol. The endorsement phase for concurrent block validation and conflict resolution, and an incentivized broadcast phase that ensures reliable delivery. While sBRB [85] provides robust Byzantine fault-tolerant dissemination guarantees, it assumes participants are willing to relay messages altruistically, which is an impractical expectation for resource-constrained mobile nodes. PocketChain addresses the lack of incentives in sBRB by embedding it within a cryptoeconomic incentive framework, where participants earn rewards for honest participation and risk penalties for malicious behavior.

4.1.2 Scalability and Lightweight Clients

Scalability in resource-constrained environments has driven research into lightweight clients and off-chain solutions. State channels [166] and rollups [29] reduce the on-chain load by batching off-chain transactions but require high availability and persistent connectivity which mobile nodes cannot always meet. Light clients, like FlyClient [27] and NIPoPoWs [106], minimize validation costs via succinct proofs but depend on full nodes for proof delivery, introducing trust and bottleneck risks. In contrast, PocketChain avoids external proof servers or persistent connectivity by design. Its parallel block construction model allows mobile nodes to contribute directly to validation using lightweight local operations and compact state representations. The scalable consensus and incentive-compatible broadcast primitives ensure that all nodes can validate and disseminate data without relying on external resource-extensive infrastructures, making it inherently more compatible with intermittent and heterogeneous mobile environments.

4.1.3 Mobile-First Architectures

Blockene [164] adopts a split-trust model inspired by Algorand [77], separating *citizen* (mobile) and *politician* (server) nodes to offload computation. While this enables mobile participation, the reliance on fixed untrusted servers undermines system decentralization and robustness in adversarial settings. Mneme [40] uses a DAG with dual consensus using a *Proof-of-Context* and *Proof-of-Equivalence* to support parallel transaction validation. However, its dependence on contextual data and complex mobile state management limit its scalability and reliability. 5G-Enabled PoA Chains [115] leverages URLLC yet depend on telecom operators as trusted validators, creating centralized trust and potential jurisdictional issues.. Instead, PocketChain embraces a decentralized, infrastructure-independent architecture in which heterogeneous nodes (e.g., smartphones, tablets, embedded devices) can fully participate via low-power, bandwidth-efficient protocols. The $\mathscr{P}ccp$ consensus mechanism avoids centralized control and heavy-weight computation by dynamic role transition and parallel block validation, thus maintaining

fair participation framework enabling any node, meeting baseline requirements to submit and validate transactions.

4.2 System Model

PocketChain design adopts standard assumptions from classical Byzantine fault-tolerant (BFT) literature regarding network synchrony, authenticated point-to-point communication links, and a static adversary controlling a bounded fraction of nodes. In addition, we introduce a transmission model for modern mobile networks, to account for energy consumption, bandwidth constraints, and heterogeneous device capabilities.

4.2.1 Network Setting

We conceptualize PocketChain as a permissionless peer-to-peer network comprising diverse mobile nodes (e.g., Smartphones, tablets). PocketChain operates under a semi-asynchronous message-passing model. To manage network latency and ensure reliability in message delivery, we consider an upper bound d on the point-to-point message delay to ensure reliable delivery within a finite time. Formally:

$$\forall m \in M,\ \forall (n_i, n_j) \in N \times N,\ \tau(m_{ij}) \leq d$$

where M is the set of messages, N the set of nodes, and $\tau(m_{ij})$ is the delivery time of message m sent from node n_i to node n_j.

The network topology is maintained through a probabilistic sampling mechanism, implemented via a peer sampling oracle Ω. At each execution, Ω generates parameterized subsets of nodes from the participant population. Each node in the network maintains direct access to this sampling service to establish its peer connections.

The PocketChain network comprises two primary components: clients, denoted as the set $\mathscr{C}$, who initiate transactions, and active network nodes, represented by the set N. Active nodes are categorized into two subsets: *announcers* $\mathscr{A}$ and *processors* $\mathscr{P}$, such that $N = \mathscr{A} \cup \mathscr{P}$ with the current number of announcers is denoted $a = |\mathscr{A}|$. Announcers are responsible for collecting transactions, forming blocks, and participating in consensus, while processors validate blocks through consensus verification. To maintain system efficiency and scalability, the number of announcers is dynamically adjusted based on stake distribution and real-time network congestion, with a target size $T_{\mathscr{A}} \approx \sqrt{|N|}$ to ensure optimal performance and load balancing.

The sets $\mathscr{A}$ and $\mathscr{P}$ are not necessarily disjoint, as nodes can simultaneously serve both roles or transition between them based on their resource availability and network conditions. Additionally, clients ($\mathscr{C}$) can evolve into active participants by joining either set $\mathscr{A}$ or $\mathscr{P}$ to contribute to consensus and earn rewards through participation in the protocol.

4.2.2 Communication links

We assume that any message m sent from a node n_i to a node n_j is delivered exactly once, although network retries may be necessary to achieve this. All traversing messages are authenticated to ensure the *no creation* property, which prevents unauthorized nodes from forging messages in the network. Formally,

if any node n_i delivers a message m, there must exist a sender s that originally broadcast m. The message m is digitally signed by the sender using their private key, and the recipients verify the signature using the sender's public key. This ensures messages cannot be spontaneously created. Finally, we assume the link abstractions as a simplified representation of communication channels that handle network-layer complexities like packet loss and latency. These abstractions are implemented through the underlying transport protocols (e.g., TCP, UDP) and adapt to heterogeneous mobile network standards (e.g., 4G, WiFi) and quality-of-service configurations.

4.2.3 Transmission model

Each participating node n_k transmits blockchain messages (e.g., transaction submission, block proposals, consensus updates) with a transmission power P_k during a designated time slot. To model the achievable data rate and energy cost of these transmissions, we adopt the classical additive white Gaussian noise channel framework and standard path-loss attenuation.

The theoretical maximum data rate over a noisy channel r_k (bits/s) for node n_k over an allocated bandwidth B_k is given by the Shannon-Hartley theorem [47]:

$$r_k = B_k \log_2(1 + SNR_k) \quad \text{with } SNR_k = \frac{P_k d_k^{-\alpha}}{\sigma^2} \tag{4.1}$$

where d_k is the distance to the receiving node, α is the environment-dependent path-loss exponent and σ^2 is the power spectral density of the Gaussian noise [81]. We modeled signal attenuation with the power-distance law $P_k d_k^{-\alpha}$. When node n_k transmits a message of size γ_k (bits), the required transmission time T_k follows directly from throughput:

$$T_k = \frac{\gamma_k}{r_k}$$

Assuming continuous transmission at power P_k, the energy consumption for uplink transmission E_k^{UL} (joules) to send γ_k bits follows the energy-per-bit model [157], defined as:

$$E_k^{UL} = T_k \times P_k = \frac{P_k \gamma_k}{r_k}$$

The physical transmission model is not required for the logical correctness of PocketChain. It is used to estimate communication cost and energy consumption under wireless/mobile networking conditions. Correctness relies on authenticated reliable links, bounded Byzantine participation, and the assumptions of the endorsement and broadcast protocols.

4.2.4 Adversarial Model

We assume a maximum of f Byzantine nodes, defined as nodes that may arbitrarily deviate from the protocol. We consider a system that tolerates up to f Byzantine faults among the active nodes N such that $f < \frac{N}{3}$. Further, the model operates under conventional cryptographic assumptions about adversarial capabilities. Specifically, we assume that the adversary lacks the ability to subvert cryptographic primitives, such as forging digital signatures.

4.2.5 Execution Model

A protocol execution proceeds as a sequence of block-validation instances. During one validation instance, the active membership is fixed: nodes may join, leave, or change roles only between two consecutive validation instances. Each block-validation instance consists of two sub-protocols: an endorsement instance among announcers, followed by a Byzantine reliable broadcast instance among processors. Multiple broadcast instances may execute concurrently, but each is mapped to a distinct virtual broadcast channel so that messages belonging to different blocks are not mixed.

4.3 PocketChain Overview

PocketChain introduces a lightweight highly scalable blockchain framework designed for efficient transaction transfer. Transactions in PocketChain are initiated by clients who possess sufficient *tips* to cover transaction fees. The PocketChain network separates active nodes into *announcers* and *processors* each fulfilling complementary responsibilities to balance scalability, liveness and Byzantine fault tolerance.

- Announcers $n_{\mathscr{A}} \in \mathscr{A}$ act as block proposers and propagators. They aggregate transactions, form candidate blocks and filter inconsistent transactions to ensure only non-conflicting candidate blocks advance. In addition to proposing blocks, announcers actively endorse valid candidate blocks proposed by others, allowing them to proceed rapidly to the broadcast phase and reach processors for final consensus. An illegible announcer $n_{\mathscr{A}}$ is a node that satisfies two conditions: (i) $n_{\mathscr{A}}$ has deposited a minimum stake $stake_{base}$ in $\mathscr{PC}$ tokens. (ii) $n_{\mathscr{A}}$ must publish a unique, cryptographically verifiable identity and advertise a reachable network address $\mathsf{addr}(n)$ (e.g., IP and port, or a peer-sampling identifier) for peer communication. After the initial advertisement, the announcer must prove that it remains online and still owns the same key or address. To do so, it sends a signed "heartbeat" message every $\Delta_{heartbeat}$ seconds (e.g., a signed timestamp, or the block IDs endorsed by $n_{\mathscr{A}}$ within $\Delta_{heartbeat}$) ensuring that $n_{\mathscr{A}}$ is currently reachable by other nodes.
- Processors $n_{\mathscr{P}} \in \mathscr{P}$ serve as consensus validators and state maintainers. Each processor maintains a consistent local view of the blockchain state and engages in the final broadcast phase of the consensus. This ensures a scalable and secure commitment mechanism, providing resilience against adversarial behavior, and guaranteeing that all honest nodes converge to the same ledger state.

As illustrated in Figure 4.2, transactions flow from clients via announcers to processors in a phased workflow according to their capabilities and incentives. This architecture allows PocketChain to decouple fast, parallelized activities of block proposal and pre-consensus, handled by announcers from the Byzantine-resilient finalization phase handled by processors, achieving horizontal scalability without compromising safety.

Upon generating a transaction, a client assigns a *tip* $\mathscr{T}$ that serves as the transaction fee and sends the transaction to multiple announcers within the network. Specifically, the transaction is sent to up to q different announcers, with q denoting the redundancy factor, to ensure fault tolerance. Transactions with fees below a $\mathscr{T}_{base}$ value are rejected, preventing spam but potentially excluding very low-value transactions. The client splits the proposed fee among the q announcers in a disproportionate manner. The first announcer receives the largest portion of the fee, with subsequent announcers receiving progressively smaller shares. Even if some announcers prioritize higher-fee transactions, others with less congestion may include lower-fee transactions. Furthermore, we suggest that if a transaction experiences excessive delay, the client can retransmit it to a smaller subset of announcers (i.e., reduce q). Importantly, only the

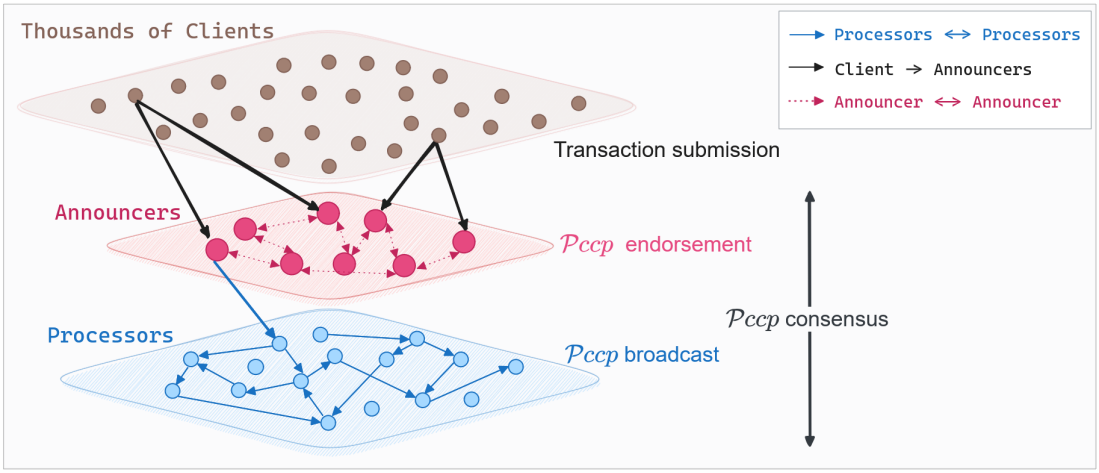


Figure 4.2: PocketChain network structure. At the top, thousands of clients initiate transactions. Transactions are submitted to a layer of announcers (shown in pink), who validate transactions and compete to form candidate blocks. Announcers also engage in an endorsement phase, followed by a broadcast phase involving processor nodes (shown in blue), to achieve final consensus.

announcer who successfully adds the transaction to the blockchain will receive their allocated share of the fee while the remaining shares are burnt.

These announcers gather the incoming transactions in their mempools, where they await validation. Each announcer receives transactions from multiple clients, each with different fee amounts due to the splitting mechanism. The announcers rigorously verify the validity of these transactions, including signatures and a sufficient account balance.

When forming a new candidate block, announcers first ensure that they have reached the *satisfactory tips threshold* $\mathscr{T}_{\text{sat}}$, a parameter that represents sufficient accumulated transaction fees for a candidate block to be submitted. To account for network congestion, we define $\mathscr{T}_{\text{sat}}$ as a dynamic function of mempool utilization. Let

$$\rho = \max\left(\frac{|\text{mempool}|}{C_{\text{mempool}}}, 1\right),$$

where |mempool| is the current number of pending transactions, C_{mempool} is the soft capacity of the mempool (i.e., a logical mempool size limit). Then, the satisfactory tip threshold is defined as:

$$\mathscr{T}_{\text{sat}} = \rho \cdot |B| \cdot \mathscr{T}_{\text{base}}$$

with $|B|$ refers to the number of transactions per block. Under light load ($\rho = 1$), announcers propose blocks as soon as the cumulative tips reach the minimum threshold. During congestion ($\rho > 1$), $\mathscr{T}_{\text{sat}}$ increases proportionally, requiring higher aggregate fees before block proposal. Valid transactions are selected for inclusion in the next candidate block, prioritizing high-fee transactions. Low-fee transactions indicate either that their originators have prioritized other announcers or that they lack prioritization. The announcers then concurrently announce their candidate blocks to all available announcers for consensus.

To achieve consensus, PocketChain employs a two-phased consensus protocol, denoted PocketChain Consensus Protocol ($\mathscr{P}ccp$). The consensus protocol consists of an *endorsement* phase primarily driven

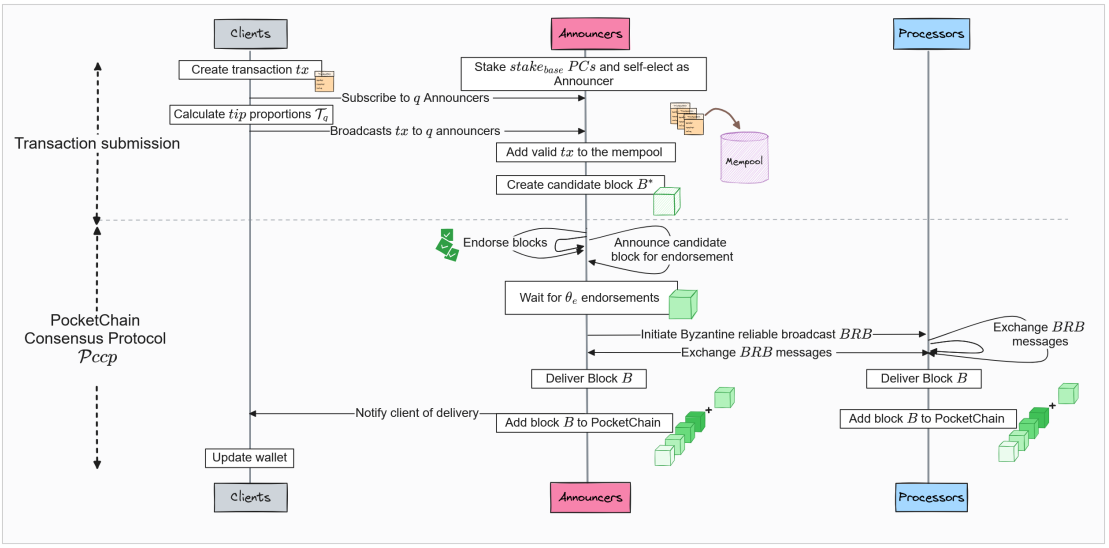


Figure 4.3: Transaction submission and PocketChain Consensus Protocol $\mathcal{P}ccp$ workflow. Clients initiate transactions, which are broadcasted to Announcers. Announcers create candidate blocks and seek endorsements from other Announcers. Once a block receives sufficient endorsements, it is announced and broadcast to the network. Processors participate in the Byzantine reliable broadcast protocol to validate and add the block to the blockchain.

by announcers who form and announce their candidate blocks in parallel for validation, followed by a *broadcast* phase where processors take over to ensure network-wide consensus and block finality.

During endorsement, announcers cross-validate each other's candidate blocks through an *endorsement* mechanism. This intermediary phase ensures the consistency of the blockchain and improves its performance by detecting and resolving conflicts at an early stage. When multiple conflicting candidate blocks reach the *endorsement* phase concurrently due to race conditions, announcers select the candidate block that has accumulated the highest number of endorsements and discard all conflicting candidate blocks. For each candidate block that passes the endorsement validation checks, an announcer signals its approval by generating a digital signature of the block hash, which serves as a single endorsement for that candidate block. Once a candidate block gathers enough endorsements from the majority of announcers, it is considered confirmed and ready to undergo the final consensus phase.

The second phase of PocketChain's consensus protocol involves processors, which constitute the majority of active nodes in the network. Processors operate with minimal restrictions, since their primary tasks of message dissemination and basic block verification demand minimal computational resources. During the $\mathcal{P}ccp$ *broadcast* phase, processors collaborate through a cost-effective broadcast primitive known as scalable Scalable Byzantine Reliable Broadcast (sBRB) [85]. Instead of requiring communication with a quorum (typically a majority of nodes), sBRB utilizes stochastic sampling. Nodes randomly select a small subset of peers for interaction, significantly reducing communication overhead and enhancing scalability. To further optimize performance, PocketChain enhances the Byzantine Reliable Broadcast protocol by adopting virtual broadcast channels (*VBChs*) [64]. Each broadcast block is assigned to a dedicated *VBChs*, which processes all broadcast messages related to that block. This parallel routing prevents interference between broadcast messages from distinct blocks, allowing independent sBRB instances to operate simultaneously.

The shift to a probabilistic broadcast model supports PocketChain's scalability in resource-limited settings. While traditional BFT mechanisms offer strong guarantees of safety and liveness, they often incur high communication overhead, limiting their practicality in large-scale or bandwidth-limited networks [38, 35]. The sBRB primitive addresses this challenge by relaxing the strictness of these guarantees, admitting a small, predefined probability of violating certain BFT properties. However, crucially, these probabilities are made arbitrarily small through careful parameter tuning [85].

Once a node receives a block and completes the *broadcast* phases, it autonomously reaches consensus. This concept is referred to as "consensus number one" [86], which posits that the delivery of a single block by any honest node is sufficient for the network to reach consensus. The broadcast primitive ensures that all honest nodes agree on the same block, even in the presence of Byzantine faults. At the end both processors and announcers keep updated copies of the ledger upon the inclusion of a validated block.

In essence, Figure 4.3 illustrates the flow of the protocol from transaction submission to the inclusion of blocks into the blockchain. The partitioned workflow ensures scalability; Announcers resolve conflicts and filter transactions early, allowing processors to focus on lightweight Byzantine consensus. By operating on disjoint failure levels, announcers prioritize liveness during transient network conflicts, and processors guarantee safety through Byzantine agreement. The $\mathscr{P}ccp$ consensus features resource-constrained, incentivized participation. Active nodes earn tips and $\mathscr{PC}$ rewards proportional to their contributions (e.g., successful block endorsements or message relaying) while balancing transmission energy costs (E_k^{UL}) and storage overhead. The transmission model directly supports incentive alignment and resource scheduling, ensuring high-participation behavior remains sustainable on constrained devices. Notably, PocketChain accommodates heterogeneous capabilities within the same class of nodes. Low-power mobile nodes might announce and endorse or validate a single block, earning rewards corresponsing to their effort, while high-resource nodes maximize earnings by engaging fully. This flexibility ensures fair inclusion across the device spectrum.

4.4 The PocketChain consensus protocol $\mathscr{P}ccp$

The PocketChain consensus protocol ($\mathscr{P}ccp$) presents a scalable consensus mechanism for heterogeneous blockchain networks. The protocol achieves efficient consensus through two complementary phases: an *endorsement* phase for initial block validation, followed by a *broadcast* phase for network-wide agreement. $\mathscr{P}ccp$ implements a two-phased approach to achieve efficient consensus, and enables optimized participation with reduced communication overhead.

4.4.1 $\mathscr{P}ccp$ endorsement

The $\mathscr{P}ccp$ *endorsement* phase marks a transition from the initial exchange of raw transactions to the preparation of block proposals for dissemination. PocketChain leverages parallel block processing, where multiple announcers concurrently propose candidate blocks aiming to optimize transaction throughput and confirmation latency. The inherent concurrency in block creation requires a robust consistency mechanism implemented through an *endorsement* process, where each candidate block B_i^* must accumulate a threshold of θ_e endorsements from distinct announcers to be considered valid and ready for the *broadcast* phase.

The threshold θ_e is configured as a quorum to balance security and liveness, ensuring Byzantine fault tolerance. The threshold θ_e is bounded by the total number of announcers, (i.e. $\theta_e \leq a$ with $a = |\mathscr{A}|$).

To tolerate up to f Byzantine announcers, we require:

$$\theta_e \geq \left\lceil \frac{a+f+1}{2} \right\rceil, \tag{4.2}$$

which ensures an honest majority even if f nodes are malicious. The lower bound on θ_e is chosen to guarantee quorum intersection. Let Q_1 and Q_2 be two endorsement quorums of size θ_e among a announcers. Their intersection satisfies $|Q_1 \cap Q_2| \geq 2\theta_e - a$. To prevent two conflicting candidate blocks from both obtaining endorsement certificates supported by disjoint honest quorums, the intersection must contain at least one honest announcer. Since at most f announcers are Byzantine, we require $2\theta_e - a > f$. Therefore, $\theta_e > \frac{a+f}{2}$, which yields $\theta_e \geq \left\lceil \frac{a+f+1}{2} \right\rceil$.

A lower value of θ_e accelerates block finalization and improves responsiveness, at the expense of weaker security guarantees, making the system more susceptible to Sybil attacks and announcer collusion. Conversely, a higher θ_e enhances resilience against adversarial behavior by requiring more endorsements for finality, though it introduces increased latency and potentially reduced availability under partial network failure.

The *endorsement* ensures that only the most robust and conflict-free blocks progress to the next stage, effectively reducing the likelihood of conflicts and improving the efficiency of subsequent consensus steps. Technically, each announcer $n_{\mathscr{A}}$ performs several checks on a received candidate block B_i^*: (i) transaction validity, where it verifies that all transactions in B_i^* are correctly formatted and valid; (ii) ledger consistency, ensuring that none of the transactions in B_i^* have already been confirmed in the ledger; (iii) announcer eligibility, confirming that the announcer $n_{\mathscr{A}}(B_i^*)$ of the candidate block B_i^* is an authorized announcer; and (iv) conflict checks, ensuring that B_i^* has no conflicting transactions with other blocks, currently under endorsement or already endorsed following Algorithm 4.

In Algorithm 4, we present an algorithm to filter out potentially invalid or conflicting candidate blocks before being handed to the *broadcast* phase. In the absence of conflicts, announcers endorse valid candidate blocks until they reach the endorsement threshold θ_e, ensuring convergence towards a consistent view of the most supported blocks. Nevertheless, conflicts inevitably arise due to the concurrent nature of block proposals.

Upon receiving a new candidate block B^*, each announcer $n_{\mathscr{A}}$ checks for conflicts with previously endorsed blocks in set S. A conflict occurs when B^* contains transactions that overlap or conflict with transactions in other endorsed blocks. The announcer $n_{\mathscr{A}}$ retrieves the current endorsement count for B^*, denoted as $|e(B^*)|$, and compares it with the endorsement counts of other candidate blocks it has received that contain transactions conflicting with those in B^*.

If $|e(B^*)|$ is strictly greater than the endorsement count of every block in the set of conflicting candidate blocks ConflictingSet, it replaces all conflicting candidate blocks in $\mathscr{S}$ (line 17–25). Otherwise, B^* is discarded (line 28). In conclusion, blocks that accumulate the most endorsements are favored and continue to receive support until reaching the threshold θ_e.

At the end, the candidate block B_i^* is promoted to a valid block B_i ready for consensus once it accumulates sufficient endorsements $|e(B_i^*)| \geq \theta_e$, with θ_e being the threshold of required endorsements referring to the quorum among the available announcers (line 39–49). Henceforth, only blocks verified for transaction integrity, non-duplication, and non-conflict advance to the broadcast phase. This mechanism favors collective agreement on conflict-free blocks over selfish fee-driven prioritization, preventing a single high-

Algorithm 4 Endorsement Algorithm

```
1:  𝒮 ← ∅                                                          ▷ Set of endorsed blocks
2:  𝒯 ← ∅                                                          ▷ Set of transactions in endorsed blocks
3:  procedure BlockProcessing(B*)
4:      if no_conflicts(B*, 𝒮) then
5:          if B* ∉ 𝒮 then                                          ▷ New block
6:              |e(B*)| ← |e(B*)| + 1
7:              𝒮 ← 𝒮 ∪ {B*}
8:              𝒯 ← 𝒯 ∪ extract_tx(B*)
9:          else                                                    ▷ Block already in S
10:             UpdateEndorsements(B*, received_endorsements)
11:         end if
12:         broadcast(B*, endorsement_proof(B*))
13:         finalize(B*)                                            ▷ Check if block can be finalized
14:     else                                                        ▷ conflicting candidate block
15:         ConflictingSet ← get_conflicting_blocks(B*, 𝒮)
16:         C_max ← get_strongest_block(ConflictingSet)
17:         if |e(B*)| > |e(C_max)| or (|e(B*)| = |e(C_max)| and hash(B*) ≤ hash(C_max)) then
18:             for all C ∈ ConflictingSet do
19:                 𝒮 ← 𝒮 \ {C}
20:                 𝒯 ← 𝒯 \ extract_tx(C)
21:             end for
22:             𝒮 ← 𝒮 ∪ {B*}
23:             𝒯 ← 𝒯 ∪ extract_tx(B*)
24:             |e(B*)| ← |e(B*)| + 1
25:             broadcast(B*, endorsement_proof(B*))
26:             finalize(B*)                                        ▷ Check if block can be finalized
27:         else
28:             discard(B*)
29:         end if
30:     end if
31: end procedure

32: procedure UpdateEndorsements(B*, received_endorsements)
33:     local_endorsements ← |e(B*)|
34:     new_endorsements ← received_endorsements \ local_endorsements
35:     if new_endorsements ≠ ∅ then
36:         |e(B*)| ← |e(B*)| ∪ new_endorsements
37:     end if
38: end procedure

39: procedure finalize(B*)
40:     if |e(B*)| ≥ θ_e then                                       ▷ B* is ready for consensus
41:         Mark B* as ready for broadcast
42:         if announcer(B*) then
43:             Initiate 𝒫ccp broadcast phase for B*
44:         end if
45:         Wait for finality of B in the blockchain                ▷ End of broadcast phase
46:         𝒮 ← 𝒮 \ {B*}
47:         𝒯 ← 𝒯 \ extract_tx(B*)
48:     end if
49: end procedure
```

fee block from dominating if it lacks sufficient endorsements. By catching conflicts early, PocketChain minimizes the resources wasted on full consensus for invalid blocks and ensures fairness in selection.

Property 1 (Single endorsement per conflict class)**.** *An honest announcer endorses at most one candidate block within the same conflict class at a given ledger height.*

Proof. Let B_i^* and B_j^* be two candidate blocks proposed for the same ledger height. Two blocks are conflicting if they contain at least one transaction that cannot be jointly finalized with a transaction in the other block, (e.g., double-spending or duplicated inclusion). According to the endorsement algorithm 4, each honest announcer maintains a set ($\mathscr{S}$) of locally endorsed blocks and a set ($\mathscr{T}$) of transactions contained in these blocks. Before endorsing a new block, the announcer checks whether the block conflicts with the transactions already present in $\mathscr{T}$. If no conflict is detected, the block may be endorsed. If a conflict is detected, the announcer applies the deterministic conflict-resolution rule and keeps only the strongest candidate according to endorsement count and, in case of equality, the deterministic hash-based tie-breaker. Therefore, an honest announcer never maintains simultaneous endorsement support for two conflicting candidate blocks at the same ledger height. Hence, an honest announcer endorses at most one block in each conflict class. □

Lemma 1 (Conflit-freedom)**.** *No two conflicting candidate blocks can both become ready for broadcast if they gather θ_e endorsements or more. Let B and B' be two distinct conflicting candidate blocks.*

Proof. Suppose by contradiction that two distinct conflicting blocks B and B', with $B \neq B'$, are both endorsed. Then, there exist sets Q_1 and Q_2 of announcers, each of size θ_e, such that every announcer in Q_1 endorsed B and every announcer in Q_2 endorsed B'. Following the endorsement quorum intersection inequality (Expression 4.2), we have:

$$|Q_1 \cap Q_2| \geq 2\theta_e - a > f,$$

where the second inequality follows from the choice of $\theta_e \geq \lceil (a + f + 1)/2 \rceil$. Since at most f announcers are Byzantine, there exists at least one honest announcer $n_{\mathscr{A}} \in Q_1 \cap Q_2$ that signed both B and B'. Since B and B' are conflicting candidate blocks, this means $n_{\mathscr{A}}$ endorsed two distinct blocks in the same conflict class. This contradicts the Property 1, which guarantees that no honest announcer signs two conflicting candidate blocks. Therefore, no two distinct conflicting candidate blocks can both accumulate at least θ_e endorsements and become ready for broadcast. □

Theorem 1 (Convergence)**.** *The endorsement algorithm converges to a consistent state across all announcers, even in the presence of temporary inconsistencies caused by network delays.*

Proof Sketch. We proceed by establishing several key properties of the algorithm:

1. **Endorsement monotonicity:** For any candidate block B^*, its endorsement count $|e(B^*)|$ is monotonically increasing over time.

2. **Endorsement bound:** For any candidate block B^*, $|e(B^*)| \leq a$, where $a = |\mathscr{A}|$ is the total number of announcers in the network. In the absence of conflicts, each announcer $n_{\mathscr{A}}$ can endorse a block at most once. If $B^* \notin \mathscr{S}$, $n_{\mathscr{A}}$ adds B^* to $\mathscr{S}$ and increments $|e(B^*)|$ by 1. Otherwise ($B^* \in \mathscr{S}$), $n_{\mathscr{A}}$ only new (non-duplicate) endorsements are incorporated, accounting for delays or asynchrony. When conflicts occur (lines 14–25), resolve the ambiguity by discarding blocks with fewer endorsements and retaining only the strongest candidate with the highest $|e(B^*)|$. A block B^* is finalized only if $|e(B^*)| \geq \theta_e$, since $\theta_e \leq a$, the endorsement count cannot exceed a.
3. **Deterministic conflict resolution:** Given two conflicting candidate blocks B_1^* and B_2^*, the outcome of the conflict resolution is deterministic and identical across all nodes once they have the same information.
4. **Eventual consistency:** In an eventually synchronous network, there exists a time t after which all announcers have the same information about all blocks and endorsement counts.

Given these properties, we can argue for the convergence of the algorithm as follows. After a sufficient time t, all announcers will have the same knowledge about the blocks in the network. For any pair of conflicting candidate blocks, the nodes resolve the conflict in the same way, ensuring consistency in the block selection process. As the consensus progresses, the endorsement count for the winning block will increase, bounded by the number of announcers a. Eventually, for any block B^* with an endorsement count $|e(B^*)|$ exceeding the threshold θ_e, B^* will be finalized and ready for broadcast. Thus, the algorithm eventually converges to a consistent state among announcers. □

Theorem 2 (Ledger safety)**.** *At any two honest processors, the finalized blockchain prefix is conflict-free.*

Proof. Every finalized block must first satisfy the endorsement phase. Given Theorem 1, the endorsement algorithm converges to a consistent valid block. By the conflict-freedom Lemma 2, conflicting candidate blocks cannot simultaneously reach the endorsement threshold. Finally, sBRB guarantees ϵ-consistency and ϵ-totality, ensuring that honest processors deliver the same finalized blocks. Therefore, the set of finalized blocks observed by honest processors is conflict-free and consistent across the network with probability at least $1 - \epsilon$. □

Communication and Computation Complexity

In the case of no conflict, each announcer broadcasts the candidate block B^* with its endorsement proof to $a - 1$ other announcers, and each announcer will endorse the non-conflicting candidate blocks once and broadcast their endorsement. Hence, the communication complexity per announcer in the case of no conflict is $O(a)$.

The communication complexity in the case of conflicts involves the same initial broadcast as the no-conflict case, but it also requires each announcer to send additional messages related to conflict detection and resolution. The worst-case scenario refers to slow progress toward validation due to multiple conflicts between blocks in the *endorsement* phase with endorsement updates arriving sequentially. Under these

conditions, the endorsement count $e(B^*)$ for a candidate block B^* advances incrementally toward the threshold θ_e. The sequential progression requires multiple rounds of message exchanges and endorsement updates before the blocks are considered valid.

Theorem 3 (Communication Complexity)**.** *Let $a = |\mathscr{A}|$ be the number of announcers in the network. The worst-case communication complexity per announcer to reach the endorsement threshold θ_e is $O(\theta_e \cdot a)$.*

Proof. Let $a = |\mathscr{A}|$ be the number of announcers, and θ_e the endorsement threshold at time t. In the worst case, every announcer broadcasts its current state to all other announcers, with the number of messages sent per announcer being $O(a)$. As updates must occur at most θ_e times, the worst-case message complexity is $O(\theta_e \times a)$. □

Theorem 4 (Computation Complexity)**.** *Let $|B^*|$ represent the number of transactions per block, $|\mathscr{S}|$ the number of endorsed blocks, and $a = |\mathscr{A}|$ the number of announcers in the network. The worst-case computational complexity for an announcer to reach the endorsement threshold θ_e for a candidate block B^* is $O(|B^*| \cdot |\mathscr{T}|) + O(a)$.*

Proof. To ensure that B^* does not conflict with the candidate blocks endorsed previously in $\mathscr{S}$, each announcer compares each transaction in B^* with $\mathscr{T}$ (i.e., transactions of blocks in $\mathscr{S}$). Thus, the complexity of conflict detection is $O(|B^*| \cdot |\mathscr{T}|)$

Once a block is verified as free of conflicts, the announcer increments the endorsement count $e(B^*)$ for the block with a complexity of $O(1)$. However, if the announcer receives new endorsement updates from other announcers, it must compare the new endorsement count with its local endorsement count, then update the endorsement count. In the worst case, each announcer could receive as much updates as the number of announcers for the same candidate block. Thus, the worst-case complexity for processing updates is $O(a)$. Thus, the worst-case computational complexity per announcer to reach the endorsement threshold is $O(|B^*| \cdot |\mathscr{T}|) + O(a)$. □

The complexity analysis reveals nuanced scalability of $\mathscr{P}ccp$ endorsement. Notably, this complexity represents a worst-case upper bound, a scenario rarely encountered in practice. While the worst-case bound reflects sequential endorsement propagation, several practical factors significantly optimize real-world performance. Since the proportion of announcers ($|\mathscr{A}|$) within the overall network is typically small, the actual number of messages exchanged remains limited even in these extreme cases. Furthermore, the protocol benefits from batch endorsement updates and parallel processing, allowing nodes to converge much faster than the theoretical bounds suggest.

Conflicts indeed increase the complexity of $\mathscr{P}ccp$. Yet, an essential element that impacts the communication and computation overhead of $\mathscr{P}ccp$ endorsement is the conflict ratio, which varies with network conditions. During high network activity, multiple conflicting candidate blocks may reach the *endorsement* phase concurrently due to network delays and the client redundancy strategy. The higher the number of announcers that a client submits to, the greater the likelihood that overlapping transactions are included in different candidate blocks, ultimately resulting in conflicts during the endorsement stage (Lemma 2).

Lemma 2 (Transaction Redundancy and Block Conflict)**.** *Let tx be a transaction submitted to q different announcers, that represents the redundancy factor, and p the probability for tx being included in any*

candidate block proposed by an announcer. Assuming that the announcers select transactions independently, the probability of tx being included in at least two blocks is lower bounded by:

$$P_{conflict}(q) \geq 1 - (1-p)^q - qp(1-p)^{q-1}$$

Proof. The probability of tx not being included in any block is $(1-p)^q$. The probability of tx being included in exactly one block is $qp(1-p)^{q-1}$, since there are q announcers, and we need one inclusion, and $q-1$ non inclusion. Therefore, the complementary probability relationship to find a lower bound for the probability of conflict is:

$$1 - [(1-p)^q + qp(1-p)^{q-1}]$$

□

Note that active participation in the endorsement process gives announcers a real-time overview of the circulating transactions in the network. Specifically, each announcer strategically reorders and batches its candidate block to maximize its validation probability and minimize transaction contention.

Such a proactive approach to transaction selection not only reduces the conflict ratio but also shortens endorsement cycles and alleviates communication overhead. Consequently, when the system transitions to the broadcast phase, processor nodes operates on pre-validated, batched blocks in parallel instead of individual transactions as in prior sBRB -based dissemination schemes [85, 65], thereby improving throughput and minimizing wasted computation.

4.4.2 Block Broadcast

PocketChain integrates a scalable implementation of Scalable Byzantine Reliable Broadcast (sBRB) [85] as a core component of $\mathscr{P}ccp$ protocol leveraging its Byzantine fault tolerance (BFT) guarantees for network-wide agreement. The endorsement phase acts as a pre-filtering mechanism to reduce the computational load on processors and ensure only pre-validated non-conflicting candidate blocks enter the BFT broadcast phase. This design choice applies sBRB at the block level rather than to individual transactions, thereby reducing per-transaction communication overhead by a factor of $O(|B|)$, with $|B|$ being the number of transactions per block.

Once a proposed block B_i has accumulated the endorsement threshold θ_e, $\mathscr{P}$ccp transitions to a three-stage Scalable Byzantine Reliable Broadcast (sBRB) for final dissemination. sBRB serves as a foundational primitive for the second phase of $\mathscr{P}ccp$ consensus. The broadcast abstraction ensures that messages originating from valid announcers are reliably delivered by honest nodes in the network, even in the presence of adversaries. The broadcast primitive operates in three phases: message dissemination, filtering of inconsistent messages, and confirmation for network-wide reach, after which the message is delivered.

Each processor $n_{\mathscr{P}}$ maintains four dynamically sampled independent, logarithmic-sized sets of other processors. A *Gossip sample* $G_{n_{\mathscr{P}}}$ for rapid, probabilistic dissemination. An *Echo sample* $E_{n_{\mathscr{P}}}$ for consistency checks on received messages. A *Commit sample* $C_{n_{\mathscr{P}}}$ for totality confirmation and a *Delivery sample* $D_{n_{\mathscr{P}}}$ for final delivery. We distinguish for each sample $X \in \{G, E, C, D\}$ an outgoing set $X_{n_{\mathscr{P}}}$, containing the peers to which $n_{\mathscr{P}}$ sends broadcast messages, and an incoming set $X^-_{n_{\mathscr{P}}}$, containing exactly those peers that have subscribed to $n_{\mathscr{P}}$ (so $n_{\mathscr{P}2} \in X_{n_{\mathscr{P}1}}$ if and only if $n_{\mathscr{P}1} \in X^-_{n_{\mathscr{P}2}}$).

To broadcast, the block announcer, now acting as a processor, invokes $\mathsf{broadcast}(B_i)$. It sends an $\text{INIT}(B_i)$

to all members of its gossip sample $G_{n_{\mathscr{P}}}$. Upon first receipt of INIT(B_i), each processor performs validity checks on B_i, marks it as seen, and then forwards INIT(B_i) to its own $G_{n_{\mathscr{P}}}$ while simultaneously sending ECHO(B_i) to its echo subscription sample $E^{-}_{n_{\mathscr{P}}}$.

When a processor receives at least θ_{echo} ECHO(B_i) messages from its $E_{n_{\mathscr{P}}}$, it concludes that a large overlapping subset of honest peers has seen B_i. It then issues COMMIT(B_i) to its commit subscription sample $C^{-}_{n_{\mathscr{P}}}$. Upon collecting θ_{commit} distinct COMMIT(B_i) messages in $C_{n_{\mathscr{P}}}$, a processor sends COMMIT(B_i) to further amplify agreement. In parallel, it counts commits that arrive from its delivery sample $D_{n_{\mathscr{P}}}$, and once it has $\theta_{delivery}$ of commits, it triggers deliver(B_i). These three phases guarantee that all honest processors agree on and output the same block, while the per-node communication and storage remain $O(\log n)$. Algorithm 5 outlines the core mechanisms of the sBRB protocol. For detailed specifications and formal analysis, we refer to the work of Guerraoui et al. [85].

The core strength of $\mathscr{P}ccp$ lies in its sampling strategy. Using sample sets of logarithmic size instead of broadcasting to large quorums. The logarithmic bound is inherited from the scalable BRB primitive. Importantly, these samples are designed to statistically reflect the overall state of the network, providing resilience against Byzantine actors. We refer the reader to the comprehensive details and proofs in [85].

Algorithm 5 Scalable Byzantine Reliable Broadcast (sBRB)) [85]

function BRB_BROADCAST(B_i)
  broadcast ⟨INIT, B_i⟩ to all $p \in G_{sender}$ ▷ Sender broadcasts block B_i
end function

Upon receiving⟨INIT, B_i⟩ from $G_{receiver}$
if valid(m) **and not** received[B_i] **then**
  received[B_i] ← **true**
  send ⟨INIT, B_i⟩ to all $p \in G_{receiver}$ ▷ Forward to one's gossip sample
  send ⟨ECHO, B_i⟩ to all $p \in E^{-}_{receiver}$
end if

Upon receiving⟨ECHO, B_i⟩ from $E_{receiver}$
if *count_echo*(B_i) $\geq \theta_{echo}$ **and not** echoed[B_i] **then**
  echoed[B_i] ← **true**
  send ⟨COMMIT, B_i⟩ to all $p \in C^{-}_{receiver}$
end if

Upon receiving⟨COMMIT, B_i⟩ from p
if $p \in C_{receiver}$ **then**
  replies_ready[B_i].*add*(p)
end if
if $p \in D_{receiver}$ **then**
  replies_delivery[B_i].*add*(p)
end if
if |*replies_ready*(B_i)| $\geq \theta_{commit}$ and not committed[B_i] **then**
  committed[B_i] ← **true**
  send ⟨COMMIT, B_i⟩ **to** p in $C^{-}_{receiver}$
end if
if |*replies_delivery*(B_i)| $\geq \theta_{delivery}$ **and not** delivered[B_i] **then**
  delivered[B_i] ← **true**
  deliver(B_i) ▷ Final message delivery
end if

To increase parallelism, we adopt a routing mechanism using virtual broadcast channels (*VBChs*), inspired

by the LightTx protocol [65, 64]. Each *VBCh* is uniquely identified and directs the flow of messages related to a specific block broadcast enabling concurrent sBRB broadcast instances without cross-channel interference. Nodes maintain a dynamic routing table that maps ongoing broadcast instances to their corresponding *VBCh*s where all the related messages to the block circulate. Each ongoing broadcast of a block B_i will correspond to an entry in the routing table that routes all the messages corresponding to the validation of block B_i. For a broadcast message m corresponding to a block B_i, the message m is routed following the predicate:

$$\mathit{Propagate}(m, B_i) : \text{if } m(B_i), \text{ route } m \text{ via } R[B_i]$$

To optimize resource utilization, the system automatically cleans up completed broadcasts. When the block B_i is successfully delivered, its associated $VBCh_i$ is removed from the routing table:

$$\mathit{Terminate}(B_i) : \text{if } \mathit{deliver}(B_i),\ R[B_i] \leftarrow \mathit{delete}(VBCh_i)$$

The implementation of these *VBChs* enables parallel propagation and validation of multiple blocks on separate virtual channels, which reflects a higher throughput.

4.5 Incentives and Resource Optimization

4.5.1 Incentives and Tips Dynamics

PocketChain introduces a dual token incentive model designed to address the specific challenges of blockchain in resource-constrained environments. Separating transaction fees (*tips*) from $\mathscr{PC}$ native tokens mainly responds to two critical requirements: first, it enables efficient micropayments where traditional high-fee structures may be prohibitive, and second, it establishes robust barriers against Byzantine behavior in environments with reduced computational security guarantees.

The *tips* $\mathscr{T}$ function as essential credit points for client nodes to submit transactions. Primarily, *tips* represent the only fee form that allows a client to submit a transaction to the PocketChain network, even when holding sufficient $\mathscr{PC}$ coins to transfer. For any transaction tx submitted by a client c, the protocol requires a tip $\mathscr{T}$, which is distributed among the q announcers to whom the client subscribes. The distribution of the tip creates proportional rewards to ensure that different announcers prioritize different transactions and avoid overlapping transactions in their blocks. Furthermore, the random allocation of fees prevents any single announcer from consistently receiving the highest rewards, promoting a fair distribution of transaction inclusion opportunities. To distribute the tip, a client first generates k proportions according to an exponential decay factor $\alpha \in [0, 1]$:

$$p_k = \{\alpha^k : 0 \leq k \leq q, \alpha \in [0, 1]\}$$

These proportions determine how the proposed tip is distributed, with earlier announcers receiving larger shares. The proportions are then normalized to ensure the sum of the proportions is equal to 1:

$$p = \left\{ \frac{p_k}{\sum_i p_i} : 0 \leq k \leq q \right\}$$

The proposed tip $\mathscr{T}$ is distributed across q announcers, with the tip for each announcer calculated as:

$$\mathscr{T}_k = \begin{cases} \mathscr{T} \cdot p_k & \text{if } \mathscr{T}^k \geq \mathscr{T}_{base} \\ 0 & \text{otherwise} \end{cases}, \quad 0 \leq k \leq q$$

Here, each announcer receives at least the minimum tip threshold $\mathscr{T}_{base}$. If an announcer's calculated share of the tip is less than $\mathscr{T}_{base}$, they do not accept to process the corresponding transaction. Notably, the minimum tip $\mathscr{T}_{base}$ prevents network flooding by low-fee transactions that offer insufficient compensation for announcers.

Clients can acquire *tips* via a direct purchase from existing holders or by serving as processor nodes in the network, since every processor earns micro-rewards *tips* whenever it successfully relays or commits a block. The system incentivizes this network participation through microrewards in the form of tips, creating a self-sustaining cycle where active contribution leads to tip accumulation.

As announcers engage in a more demanding role in PocketChain network, namely forming, announcing, and endorsing new blocks. Primarily, the announcer who successfully adds the transaction to the blockchain and creates a valid block will receive their allocated share of the associated tips. Additionally, other announcers who contribute to the process by endorsing blocks, resolving conflicts, and maintaining an updated state of the network are rewarded with $\mathscr{PC}$ coins upon successful block inclusion in the blockchain ledger. Announcers are required to deposit a stake in the form of $\mathscr{PC}$ tokens. All announcers are subject to uniform stake requirement $stake_{base}$. By enforcing stake equality, PocketChain avoids centralization risks that can arise in some PoS systems where voting power or block proposal probability correlates with stake size.

The stake in PocketChain serves as a sybil-resistance bond and a barrier to malicious behavior; any announcer engaging in Byzantine behavior such as proposing spam or invalid blocks, risks slashing their stake, thereby incentivizing honest participation. To dynamically balance endorsement redundancy and network overhead, PocketChain regulates announcer participation through reward-driven stabilization. The system targets $T_{\mathscr{A}} \approx \sqrt{|N|}$ active announcers, scaling sublinearly with network size. The target $T_{\mathscr{A}}$ is a scalability heuristic that balances two opposing costs. If the announcer set is too small, transaction intake becomes a bottleneck and the system becomes more vulnerable to temporary announcer failures. If the announcer set is too large, the endorsement phase becomes expensive because endorsement communication grows with the number of announcers. The announcer reward mechanism is designed to naturally stabilize around the target number of active announcers $T_{\mathscr{A}}$.

$$\mathscr{B}(a) = \begin{cases} \mathscr{B}_{\max}, & a \leq T_{\mathscr{A}} \\ \mathscr{B}_{\max} \dfrac{T_{\mathscr{A}}}{a}, & a > T_{\mathscr{A}} \end{cases}$$

where $\mathscr{B}_{\max}$ is the total $\mathscr{PC}s$ budget per block when $a \leq T_{\mathscr{A}}$. Each announcer that contributes to a finalized block receives an equal share $\mathscr{R} = \frac{\mathscr{B}(a)}{a}$, regardless of stake size. This approach avoids parameter proliferation, ensures fairness, and incentivizes a balanced number of announcers aligned with system scalability.

The protocol establishes a critical security threshold to ensure that the potential slashing exceeds any benefits from Byzantine behavior by setting $stake_{base} \gg max\_rewards(t)$, where $max\_rewards(t)$ represents the maximum attainable rewards through legitimate announcing activities over a time period t.

4.5.2 Storage Reduction Strategy

As the blockchain grows, the challenge of bootstrapping new nodes becomes critical. New nodes, in particular, need to efficiently acquire and validate historical state without storing the complete blockchain history [202], which is particularly challenging in resource-constrained mobile environments. PocketChain addresses storage constraints in mobile devices through a hierarchical summarization mechanism that combines succinct cryptographic proofs with efficient state management. We leverage Merkle paths to create verifiable commitments of block states, enabling efficient storage and retrieval while maintaining the ability to prove state integrity. Our approach exploits the power law distribution of blockchain user activity, where a small number of users generate the majority of transactions, while most remain relatively inactive [110].

The system operates through a series of summary blocks $(S_0, S_1, S_2, \ldots)$, each representing a compact commitment to the state of user balances for a set of blocks through a Merkle root. Subsequent summary blocks recursively incorporate both the state captured in their predecessor (S_{i-1}) and the user balance updates resulting from the next set of transactions. As summaries progress, the likelihood of summarizing transactions from the same users increases. This results in the aggregation of balances belonging to active users into newer summaries, leaving older summaries to primarily track inactive users. After several iterations, a more compact representation is implemented by hierarchically aggregating multiple summaries into larger units, referred to as $xBlocks$, spanning vast numbers of transactions.

State retrieval is facilitated by traversing the hierarchy of summary blocks, starting from the most recent $xBlock$ and proceeding backward through previous summaries. Following the *power law distribution* of blockchain activity, active users (a small subset) continue to generate new transactions, so their balances are updated in newer summaries, while inactive users remain in older summaries, which are accessed less frequently or archived over time. In this case, information related to inactive users may eventually be moved to archival storage or removed entirely, depending on the system's data retention policy.

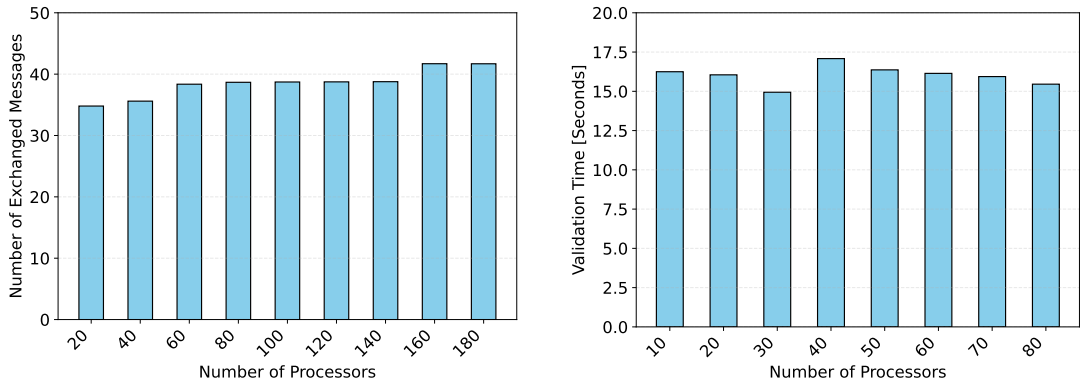
Our storage optimization solution is currently under active development, and we are working on formalizing its properties and security guarantees. Future work will present a comprehensive analysis of the theoretical foundations and practical implementations of the mechanism.

4.6 Experiments

In this section, we present a comprehensive set of experiments to evaluate the efficiency of PocketChain under various network conditions and configurations. Our analysis focuses on two key dimensions: performance and scalability. We measure critical metrics including time and communication overhead, latency and throughput, and consensus energy consumption, highlighting the potential of PocketChain to support even the most resource-constrained devices. Below, we detail our experimental setup, methodology, and our key findings.

4.6.1 Experimental Setup

Implementation Detail. To evaluate PocketChain, we developed a comprehensive prototype implementing its full range of functionalities. The source code, written in Python 3.10, is designed for portability and can be readily implemented on iOS and Android mobile devices using the cross-platform library Kivy. The source code, along with detailed configuration files and replication instructions, is available at



(a) Messages exchanged for validating a block with varying number of processors.

(b) Consistent validation latency for a PocketChain block within $\mathscr{P}ccp$ broadcast phase.

Figure 4.4: Communication and time complexity of the broadcast phase in the PocketChain Consensus Protocol $\mathscr{P}ccp$.

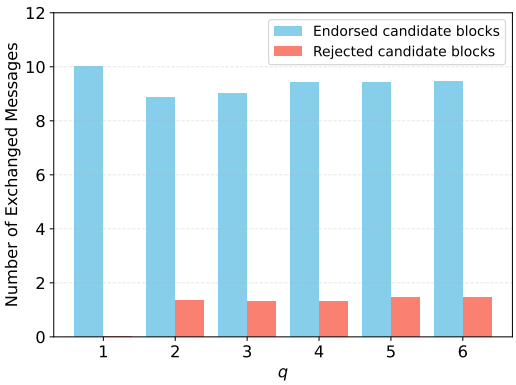
https://github.com/ImaneElabid/PocketChain.
For ecosystems prioritizing continuous high throughput, we recommend deploying PocketChain on mid-range to high-end smartphones or tablets equipped with multicore ARM CPUs (e.g., Cortex-A510 at ≥1.8 GHz) with integrated hardware cryptographic accelerators and enough free storage to host a local copy of the blockchain in order to ensure optimal performance and security. For network connectivity, the system is optimized for modern wireless standards, with optimal operation over 4G/5G cellular or Wi-Fi connections to minimize latency and maximize throughput for peer-to-peer block propagation and consensus messages. However, the protocol imposes no minimum participation threshold. Lower-profile nodes with constrained resources, such as older mobile devices, can still participate by validating even a single block. All experiments were simulated on a physical workstation running Ubuntu 20.04 LTS, with an Intel Xeon W-2123 quad-core processor (3.6 GHz base frequency) and 32 GB of memory.

Nodes Distribution Our experiments were conducted on a simulated network of 200 mobile devices with varying computational and network capabilities, reflecting the different roles in PocketChain system. Unless otherwise specified, a total of 2000 transactions were submitted to the network, grouped by announcers into blocks of 50 transactions each. Each client subscribed to $q = 3$ announcers for block dissemination. Clients were required to provide a minimum tip of $\mathscr{T}_{base} = 10$ to submit a transaction to an announcer. Announcers, on the other hand, were required to stake $100\mathscr{PC}s$ each to assume the role of an announcer and begin forming and broadcasting blocks.

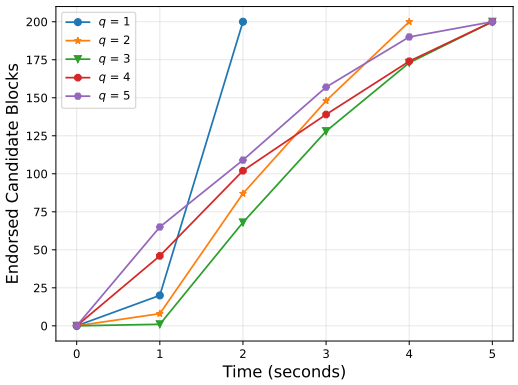
4.6.2 Evaluation Results

Communication Overhead and Latency

In this set of experiments, we evaluate the block broadcast consensus and candidate block endorsement phases of the PocketChain consensus protocol $\mathscr{P}ccp$ by examining two key metrics that directly impact efficiency: *communication overhead* and *processing latency*. Our evaluation methodology isolates each phase of $\mathscr{P}ccp$ to analyze how specific protocol parameters, such as network size and redundancy factor



(a) The impact of q on the endorsement communication complexity.



(b) The impact of q on the endorsement throughput.

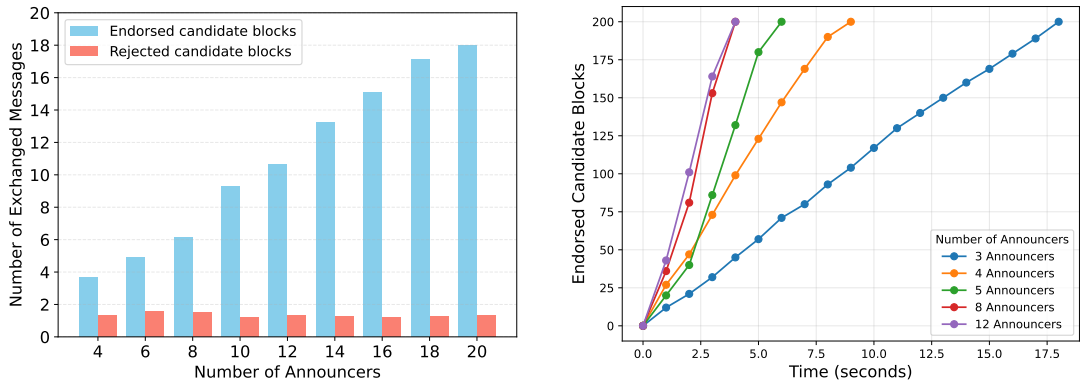
Figure 4.5: The impact of the redundancy factor q on the endorsement phase of PocketChain: Higher values of q result in more conflicts, but the endorsement phase maintains a linear communication complexity relative to the number of announcers.

q, affect system performance.

Communication Overhead. To evaluate the communication overhead during the block broadcast phase, we measured message complexity per processor while varying the network size from 20 to 180 processors. Figure 4.4a illustrates the correlation between the number of processors participating in the broadcast consensus and the number of messages exchanged to achieve block validation in $\mathscr{P}ccp$. The obtained results indicate a logarithmic increase in communication overhead as the number of processors grows. This logarithmic communication pattern stems from the underlying gossip-based dissemination and peer sampling employed by $\mathscr{P}ccp$, wherein each processor communicates with a small, randomly selected sample among active nodes, ensuring minimal communication overhead.

As more processors join the network, the samples sizes for each processor increases logarithmically, effectively limiting communication overhead. Consequently, even in large networks of active nodes, the average number of messages exchanged per node remains bounded while maintaining a highly decentralized protocol. The minimal message complexity per node, even as the network scales, is particularly significant for mobile environments, where communication efficiency directly impacts battery life and bandwidth usage.

Next, we assess the impact of the number of processors on the validation time of the blocks. Figure 4.4b shows that the validation time is minimally affected by the increase in the number of processors. This is attributed to the logarithmic communication complexity of $\mathscr{P}ccp$, which limits the number of peer interactions required for validation. The logarithmic communication of $\mathscr{P}ccp$ broadcast ensures that the validation time remains stable, even as the number of participating processors increases dramatically. In essence, the ability to deliver predictable and consistent validation times, regardless of network size, is a major property that facilitates the adoption of PocketChain by systems that require predictable latency under varying network conditions, such as resource-constrained networks. The efficiency of $\mathscr{P}ccp$ broadcast in terms of communication and time complexity, as demonstrated in these experiments, confirms the suitability of $\mathscr{P}ccp$ for large-scale decentralized networks, including resource-constrained



(a) Messages exchanged during the endorsement phase for valid and conflicting candidate blocks with varying numbers of announcers.

(b) The impact of active announcers on validated candidate blocks over time during the endorsement phase.

Figure 4.6: Communication and time complexity of the endorsement phase in the PocketChain Consensus Protocol $\mathscr{P}ccp$.

deployments.

Next, we focus on the empirical assessment of $\mathscr{P}ccp$'s *endorsement* phase in terms of performance and scalability. This phase is critical to ensure the validity and consistency of candidate blocks proposed for broadcast consensus. We first evaluate the communication overhead required to reach the endorsement threshold for a candidate block or to reject a conflicting or invalid one. In this experiment, the endorsement threshold is set to a majority, $\theta_e = \lceil \frac{2a}{3} \rceil$, where $a = |\mathscr{A}|$ is the number of announcers. Message complexity per announcer is measured for both endorsed (valid) and rejected (invalid) blocks while varying the number of announcers. Figure 4.6a presents the number of endorsement messages exchanged for both the endorsed and rejected candidate blocks as the number of announcers increases. The results confirm the conceived role of $\mathscr{P}ccp$ endorsement in detecting and filtering conflicting candidate blocks early in the endorsement phase. Regardless of the number of announcers, candidate blocks with overlapping or conflicting transactions are detected and rejected early in the endorsement phase and generate remarkably low message exchanges, ensuring minimal communication overhead. The number of messages for the endorsed blocks, on the other hand, grows linearly with the number of announcers in the network, since the endorsement threshold θ_e is proportional to the participating announcers. Each valid block must collect endorsements from a majority of announcers to achieve consensus.

The results validate two key design choices in $\mathscr{P}ccp$ endorsement mechanism: (i) the efficient early rejection of invalid/conflicting candidate blocks, which prevents unnecessary network congestion, and (ii) the controlled linear scaling for valid blocks, which ensures agreement without overwhelming the network. Since announcers represent only a subset of the total network participants, this linear communication complexity remains practical even as the network size expands.

Comparably, we evaluate the impact of the number of participating announcers on $\mathscr{P}ccp$ endorsement throughput. In this experiment, we use a fixed value of the redundancy factor $q = 3$, and measure the average number of validated candidate blocks over time with varying numbers of active announcers. The results obtained in Figure 4.6b demonstrate that increasing the number of announcers leads to a substantial increase in endorsement throughput, as multiple announcers concurrently process and validate candidate

blocks. However, significantly increasing the number of announcers may result in an extra communication overhead associated with the increase in endorsement messages, as shown in Figure 4.6a. In practice, the number of active announcers serves as an adaptive control, balancing performance and network conditions, underscoring the flexible design of PocketChain for various operational scenarios and deployment needs.

Now, we examine the impact of the redundancy factor q on the performance of the endorsement phase. In PocketChain, q represents the number of announcers that receive the same transaction from a client, directly influencing the likelihood of conflicts during endorsement.

Figure 4.5a illustrates the impact of the redundancy factor q on the endorsement communication complexity in a network of 10 announcers. The stable number of endorsement messages exchanged regardless of q indicates that transaction redundancy has very little impact on the communication complexity of the endorsement phase. Additionally, figure 4.5a distinguishes between endorsed and rejected candidate blocks. Rejected candidates blocks correspond to those containing conflicting or overlapping transactions, which are identified and discarded during the endorsement phase. The low and constant number of such rejections demonstrates the ability of $\mathscr{P}ccp$ protocol to filter out conflicting candidate blocks early, without incurring additional communication cost as q increases. This independence from q demonstrates a key scalability property of $\mathscr{P}ccp$ endorsement phase. Even as clients establish more redundant connections to announcers (higher q values), the message overhead per announcer remains bounded. Such behavior ensures that the system maintains consistent communication efficiency regardless of how widely transactions are distributed across announcers.

On the other side, Figure 4.5b illustrates the impact of redundancy factor q on the endorsement throughput. We observe that increasing q consistently reduces the system block validation rate. When q increases, each transaction is distributed to a larger number of announcers, creating more overlap in the transactions processed by different announcers. This overlap induces a higher probability of concurrent processing of the same transactions, resulting in conflicts during the endorsement phase. As PocketChain must maintain consistency, these conflicts trigger the rejection of conflicting candidate blocks, directly reducing the system's throughput. With low redundancy, transactions are distributed more sparsely across announcers, allowing for parallel processing with fewer conflicts. Conversely, higher redundancy creates dense transaction overlaps, increasing conflict probability and consequently reducing the rate of successful block validations.

To evaluate transaction inclusion latency in PocketChain, we submitted a mixed set of transactions where tips were randomly chosen between the minimum tip value $\mathscr{T}_{base} = 10$ and a 10× higher value $\mathscr{T}_{10} = 100$, and recorded the number of blocks they waited before inclusion.

Figure 4.7 shows that 52% of the transactions were immediately included (0 blocks), 39% after one additional block, and the remaining 6% after two blocks. No transaction is delayed indefinitely, with all inclusions bounded within three blocks. The bounded latency distribution demonstrates the resilience of PocketChain to starvation. These findings validate the benefit of client-side tip-splitting and announcer diversity, as well as the system's scalability under moderate congestion.

Energy Efficiency

Energy consumption measurements are derived from the transmission model introduced in Section 4.2.3 For every message transmission, the simulator computes the corresponding transmission duration according to the achieved data rate and accumulates the resulting transmission energy using Eq. 4.1. As shown in

Figure 4.8a, PocketChain demonstrates exceptional energy efficiency, maintaining this property even in large networks. Notably, increasing the number of announcers, which is crucial for mitigating network congestion, has a negligible impact on the energy footprint. In contrast, increasing the number of processors leads to higher energy consumption for block validation, revealing a trade-off between energy efficiency and decentralization. While adding processors enhances decentralization, it also increases message traffic for block validation.

Figure 4.8b further evaluates the energy efficiency of PocketChain in mobile devices and server machines under varying redundancy factor values. The results show that, by leveraging the efficient networking hardware of mobile devices, PocketChain can achieve up to 100 times greater energy efficiency compared to server machines typically used as blockchain nodes, for both announcers and processors. These findings underscore PocketChain's suitability for energy-constrained environments and its ability to outperform traditional blockchain implementations in energy efficiency.

To evaluate the benefits of PocketChain's role separation design, we conducted a comparative experiment against a traditional single-node blockchain design. In the PocketChain configuration, we delegate block proposal conflict resolution and endorsement to a small set of announcers (5 nodes) with a stake of 100 $\mathscr{PC}$ each. While enabling 100 processors to ingage in the broadcast phase for consensus finality, requiring no staking. In the single-type-node configuration, all 105 nodes participate as announcers and processors at the same time, each holding stake and conducting endorsement and generating broadcast messages. Both systems processed identical transaction volumes under equivalent network conditions.

Figure 4.9 highlights the scalability and inclusiveness advantages of PocketChain's role-separated architecture compared to single-node design. In the single-node configuration, all 105 nodes are required to stake and participate in block endorsement, resulting in significantly higher energy consumption and communication overhead, which imposes a critical barrier to decentralization. Nodes with minimal or no stake reserves cannot participate in any validation, disproportionately favoring high-resource devices and excluding low-power or resource-constrained nodes. On the other hand, PocketChain separated roles

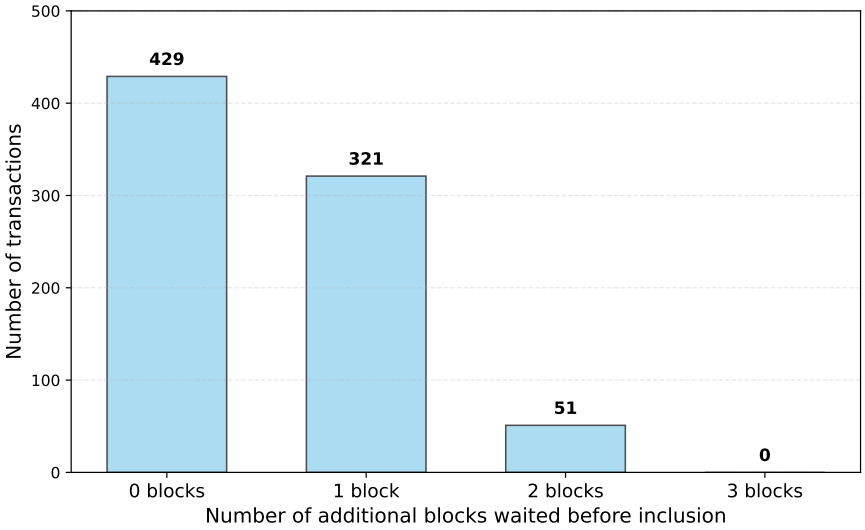
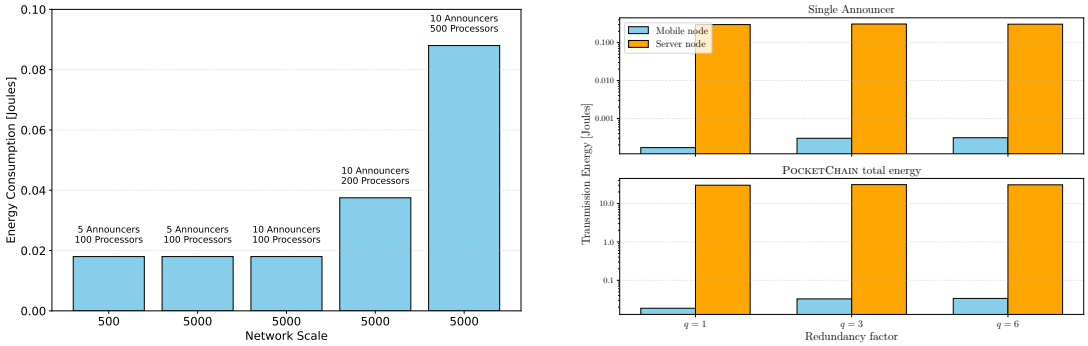


Figure 4.7: Distribution of transaction inclusion latency in PocketChain. Bars represent the number of transactions included immediately (0 blocks), after one block, two or three blocks.



(a) PocketChain's energy consumption when the network scales.

(b) PocketChain's energy consumption on mobile vs. server nodes.

Figure 4.8: Energy consumption of PocketChain consensus $\mathscr{P}ccp$. PocketChain's energy consumption is primarily influenced by the number of processors in $\mathscr{P}ccp$. Moreover, it demonstrates exceptional efficiency on mobile devices, achieving up to 100× energy savings in transmission compared to server machines.

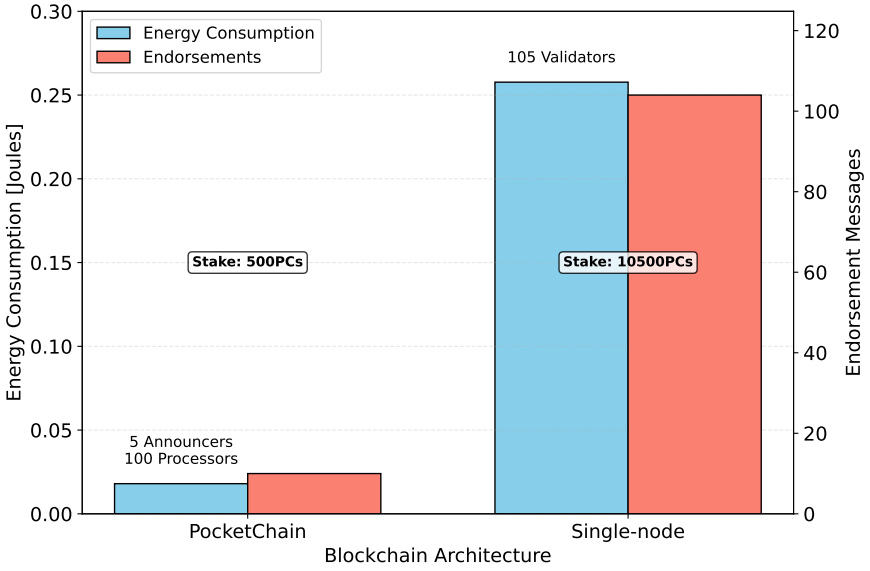


Figure 4.9: Comparison of energy consumption and endorsement message volume between PocketChain and a single-node blockchain model.

design results in about 15× reduction in energy consumption, and the number of exchanged endorsements.

Furthermore, the single-node design requires all participants to maintain full network visibility to perform endorsements securely. This global view becomes increasingly impractical in large-scale or bandwidth-constrained environments. PocketChain mitigates this issue, as only announcers require a global perspective, while processors operate with partial knowledge via small broadcast sampling.

4.7 Conclusion

PocketChain presents a significant improvement for blockchain operation on resource-constrained devices. Our analysis demonstrates that through the $\mathscr{P}ccp$ consensus protocol, PocketChain inherits $O(\log n)$ per-processor communication in the broadcast phase, while the endorsement phase scales with the announcer set size $|\mathscr{A}|$. Since $|\mathscr{A}|$ is maintained sublinear in the total network size, the combined design keeps communication practical for resource-constrained devices. The two-phase consensus efficiently filters invalid blocks during the endorsement phase before proceeding with validated block broadcast, significantly reducing network load and computational demands. PocketChain's flexible role assignment and energy-efficient consensus enable smartphones to serve as full network participants rather than mere transaction initiators. The resulting protocol eliminates hardware barriers to entry and ensures equal participation opportunities across diverse mobile devices, addressing the scalability barriers that have limited blockchain adoption in mobile environments. Additionally, our proposed hierarchical block summarization approach has significant potential to mitigate the storage load on mobile nodes and reduce bootstrapping overhead for new participants. Future work will focus on formalizing the summarization mechanism by developing rigorous proofs for state consistency and establishing optimal parameters for summary block generation, paving the way for wider adoption of blockchain technology in mobile environments.

5 Immutability Paradox: Data Redaction Benchmark

Although originally designed to support the famous Bitcoin currency [143], blockchain gained immense popularity as a potential tool for efficient data sharing and management across a range of applications, including financial services, risk management, Internet of Things, and social services [95]. The adoption of blockchain has accelerated due to the benefits it provides in terms of trustworthiness, security, and immutability among others. Although immutability is indispensable for blockchain integrity and security, this same property potentially conflicts with privacy regulations and data flexibility as sensitive information becomes permanently inscribed in the ledger [5, 136, 200, 161].

Certain use cases requiring privacy regulations, error correction, and operational flexibility, give rise to a critical question: *Can blockchain reconcile its static nature with the dynamism of real-world applications?* This tension paves the way for blockchain systems centered on context-aware mutability. In these systems, modification capabilities are neither unlimited nor arbitrary, but carefully constrained by cryptographic guarantees and governance protocols.

A novel approach promoting *redactable* or *mutable* blockchains [9] has emerged challenging the fundamental principle of immutability. The *redaction* process, renders the blockchain mutable by modifying previously recorded blockchain data as needed. Redaction is a disruptive and complex process requiring careful consideration of several factors to ensure the security and consistency of the blockchain. The emergence of redactable blockchain has gained immense interest in the blockchain community, and several follow-up studies have been conducted to investigate its potential applications and limitations.

This chapter addresses the ”immutability paradox” through two contributions: *A Comparative Analysis of Blockchain Redaction Techniques* and *Beyond Immutability: A Comprehensive Review of Redactable Blockchain Systems*. Crucially, we provide an original implementation of a comparative evaluation framework analyzing disruptive redaction methodologies, and compare their effectiveness both analytically and experimentally. Moreover, we explore the theoretical foundations on mutable blockchain architectures, covering a taxonomy of redaction techniques, implementation challenges, and security implications.

5.1 Related Work

While research specifically addressing redactable blockchains remains relatively nascent, several recent studies have begun to outline the landscape of this emerging field, offering essential initial taxonomies of redaction techniques. A study [198], offers a classification of redaction methods and briefly outline key challenges in redaction mechanism design, including a proposed set of evaluation criteria. However, it remains primarily descriptive without conducting an empirical assessment. Fathalla et al. [68] provide a high-level discussion of redactable blockchains, primarily focusing on pioneering works introducing the redaction concept through Chameleon hash functions. While they acknowledge open challenges and potential applications, their work does not offer an exhaustive survey of existing solutions and notably omits more recent advancements in the field. Additionally, while some recent studies [1, 195] are more comprehensive in addressing subsequent improvements and open challenges in the field, a granular analysis required to understand the fundamental trade-offs between flexibility, security, and the underlying design choices of various redaction techniques. The work of Heo et al. [91] explores few redaction techniques as a part of their survey about storage optimization techniques.

To position our work within the established literature, we compare our work to prior surveys on redactable blockchain systems. It is important to mention that only a limited number of studies have reviewed the concept of blockchain redaction [150, 198, 1, 68]. However, none of the prior surveys conducted an empirical comparison of the existing solutions, in contrast to what we propose in our work. To date, no prior work offers a systematic experimental benchmark of redactable blockchain systems. The lack of empirical benchmarking creates a major barrier to the informed adoption of redactable blockchain solutions. Furthermore, when new redaction techniques are proposed, authors often provide analytical results, and their solutions are rarely benchmarked against existing solutions.

Table 5.1 shows a comparison of the main features in literature with what we cover in our work. We offer a comprehensive synthesis and an experimental benchmarking framework that addresses both the practical implementation challenges of redaction techniques and their theoretical foundations. The core contributions are succinctly summarized below.

- We define a unified taxonomy of redaction techniques grounded in the core pillars of blockchain immutability: *cryptographic*, *meta-transaction*, and *consensus-based* approaches. This classification

Table 5.1: Comparison of our survey with other existing surveys.

Surveys	**[150]**	**[198]**	**[1]**	**[195]**	**[68]**	**[91]**	**Our work**
Presents benefits and limitations of redaction methods	~	✓	✓	✓	~	x	✓
Provides an updated and detailed analysis of redaction methods	x	x	x	✓	✓	x	✓
Covers subsequent improvements	x	~	~	✓	~	x	✓
Includes details about recent methods	x	x	x	x	x	x	✓
Discusses open issues	x	x	x	✓	✓	~	✓
Provides experimental benchmark	x	x	x	x	x	x	✓

x: Not addressed. ~: Partially addressed. ✓: Addressed.

reflects how each redaction technique necessarily disrupts one or more fundamental pillar of immutability, specifically, cryptographic binding, transaction semantics, and consensus. We further decompose each main class into subclasses to trace the technical evolution and operational trade-offs.

- We introduce the first experimental benchmark comparing main redaction techniques across efficiency, performance, and operational practicality. The open source implementation provides an extensible foundation for future experimentation and for integrating emerging redactable protocols.
- We outline key open challenges and outline targeted research directions, focusing on redaction latency, scalability constraints, adaptation across deployment contexts, and the need for robust integrity checks following any modification.

5.2 Background and Foundations

This section introduces the foundations of blockchain immutability based on Bitcoin's architecture as a reference (presented in 2.3.1). We then explore the key motivations driving redaction research, including regulatory compliance, ethical concerns over illicit content, and system-level optimization. Finally, we outline the essential security and performance requirements that redaction approaches must meet to ensure verifiability, transparency, auditability, and overall system integrity.

5.2.1 Immutability Mechanisms

The tamper-resistance of a blockchain rests on three interdependent pillars: cryptographic hash linking, Merkle tree validation, and decentralized consensus. Each of these mechanisms makes unauthorized modifications computationally infeasible and economically unsustainable. Below, we examine how they collectively enforce ledger immutability.

Cryptographic Hash. Cryptographic hash functions deterministically map arbitrary data to unique fixed-size outputs such that any small change in the input produces a different hash. In the blockchain, each block header includes the hash of its predecessor. The sequence of hashes connecting each block to its parent forms a chain that extends all the way back to the genesis block. Altering any block changes its hash, breaking continuity with subsequent blocks. Since all nodes maintain local copies of the chain, any user can easily verify the data integrity by comparing the freshly computed hash with the stored hash value [200].

Merkle Tree. Merkle tree is a binary hash tree used to efficiently summarize and verify the integrity of all transactions within a block. Transactions are hashed pairwise, recursively forming a tree until a single Merkle root is derived and embedded in the block header. If any transaction within the block is altered, the corresponding transaction hash changes, propagating up the Merkle tree to produce a different Merkle root. Since the block header contains the original Merkle root, this inconsistency invalidates the block and, consequently, all subsequent blocks in the chain that reference its hash. This hierarchical hashing ensures that tampering with even a single transaction is detectable, reinforcing ledger integrity [200].

Distributed Consensus Protocol. In the absence of a central authority, blockchain systems rely on distributed agreement among nodes to validate transactions and append new blocks. The decentralized consensus protocol guarantees that the blockchain remains secure and tamper-proof even in the presence of potentially malicious users or miners. In practical terms, nodes in the blockchain must reach a consensus before adding a new block to the chain. In PoW systems like Bitcoin [143], consensus is achieved through a competitive

process where miners expend computational resources to solve cryptographic puzzles. This process ties block creation to substantial computational effort, making fraudulent modifications prohibitively expensive. The high cost of computation and time required for block generation provides strong protection against tampering. Once a block is appended, any adversary attempting to alter the block must re-mine all subsequent blocks faster than the rest of the network, which becomes exponentially harder as the chain grows [172].

State Semantics. Beyond the cryptographic bindings, and consensus protocols blockchain immutability is enforced by state semantics. Fundamentally, the blockchain operates as a deterministic state machine where each transaction constitutes a state transition that moves the system from one valid global state to the next. Transactions are processed in a strict, consensus-validated order to produce a single, canonical global state (e.g., the UTXO set in Bitcoin or account balances in Ethereum) that is verifiable by all participants [144, 190]. This ordered sequence forms an append-only execution log. Any honest node executing the same sequence of transactions from the genesis block will derive an identical global state. The integrity of the current state is therefore a direct, verifiable consequence of the entire historical log. Consequently, any attempt to modify the past violates the sequential logic required to derive the current valid state.

5.2.2 Why Redact Data in the Blockchain?

The drive behind redactable blockchains stems primarily from the need to rectify data within an immutable ledger without compromising the overall integrity of the system. Software flaws, unanticipated regulatory changes, or the need to evolve business logic may require adjustments that immutable blockchains cannot accommodate [37]. Moreover, blockchain also stores arbitrary data. For instance, in Bitcoin, the `OP_RETURN` opcode allows the storage of arbitrary data as part of a transaction.[135] revealed that approximately 1.4% of Bitcoin transactions contained non-financial data, encompassing a diverse range of data types. The authors of [133] list five types of *non-financial* information that can be inserted in the blockchain and pose high risks if possessed by the users of the blockchain network, namely, copyright violations, malware, privacy violations, politically sensitive content, and illegal content. Redactable blockchains offer a solution by enabling the modification of data within an immutable ledger while preserving the overall integrity of the system. Building upon these concrete risks associated with immutable ledgers, we identify the core motivations underlying the need for data flexibility and redaction in blockchain systems.

Regulatory and Legal Requirements. Modern data protection frameworks such as the European GDPR [75] and China's PIPL [149] mandate that individuals have control over their personal data. However, traditional blockchains structurally violate this principle. Once data appear on-chain, it cannot be removed, placing blockchain deployments in direct conflict with regulatory mandates. In particular, the immutability of data in these distributed ledgers implies that one of the key principles of GDPR, Art.17 Right to erasure, is not met by blockchain, since there is no built-in mechanism to clear personal data upon request.

Ethical Considerations. The disclosure or even the mere possession of particular data, including politically sensitive material, pirated data, blasphemy, and hate speech, may be illegal in certain jurisdictions. The decentralized nature of blockchain systems can lead nodes to intentionally or inadvertently possess, distribute, or host illegal content embedded within the blockchain. The 2018 discovery of child abuse imagery encoded in Bitcoin transactions [133] exemplifies how blockchain neutrality can be weaponized, forcing participants into complicity. Once objectionable content enters the ledger, mere possession can trigger criminal liability or regulatory sanctions for node operators, even if they played no active role in its insertion. Beyond legal exposure, organizations that facilitate network participation risk serious reputational damage. Furthermore, the permanent availability of harmful content can perpetuate societal

harm indefinitely.

Technical Motivations.

Operational error correction. Complex software systems are prone to flaws; in industrial blockchain systems, errors can lead to serious disputes and may have detrimental consequences in the production chain. In sectors like healthcare or finance, erroneous data (e.g., incorrect patient records) must be rectifiable. Furthermore, in such systems, there are no third parties to rely on for assistance, and once an exploit is discovered, there is no easy method to fix it or roll it back.

Patching software flaws. Smart contracts flaws pose significant threats to Decentralized Applications (DApps) operating on the blockchain, especially in terms of monetary losses. A single vulnerability can freeze assets or siphon millions, as tragically demonstrated by the DAO attack [136]. Smart contracts cannot be patched post-deployment, leaving vulnerabilities permanently exploitable [4]. Immutable contract code cannot be patched in place without forking the network, leading to costly hard forks or fractured ecosystems. Thus, the inherent immutability of blockchain is poorly suited for applications requiring ongoing software review and the ability to update or patch contractual logic.

Storage Optimization. The append-only architecture of blockchain creates an unsustainable storage trajectory. Permanent on-chain storage is costly, inefficient, and redundant in many use cases. Much of this data represents obsolete information such as expired smart contract logs, deprecated tokens, or off-chain pointers. In 2024, for instance, the Bitcoin blockchains recorded ledger sizes exceeding 600 GB [22]. Such immense and growing storage requirements burden new nodes with limited resources, and increase the operational costs for existing participants to maintain a full copy of the ledger.

5.3 Taxonomy of Redaction Techniques

Blockchain redaction refers to the ability to modify blockchain content after it has been committed, without undermining the integrity, auditability, or security guarantees of the ledger [9]. Enabling such functionality requires relaxing one or more of the system properties that collectively enforce immutability. Building on the analysis in Section 5.2, blockchain immutability rests on three interdependent pillars, namely, *cryptographic integrity*, *consensus finality*, and *deterministic state semantics*. In this section we define a structured taxonomy of redactable blockchain techniques, derived from this perspective. Redaction techniques are classified according to the specific immutability pillar they are designed to modify. The taxonomy comprises three major classes, divided into distinct subclasses based on specific technical mechanisms and control models as illustrated in Figure 5.1: *Cryptographic Redaction*, which operates on the cryptographic integrity; *Meta-transaction Redaction*, which modifies the state semantics; and *Consensus-based (Voting) Redaction*, which enforces collective authorization through consensus.

This taxonomy is derived from a comprehensive review of the literature, tracing the evolution of redactable blockchain systems from the seminal work of Ateniese et al. [9] in 2017 to peer-reviewed advancements published through 2025. Our analysis prioritizes works that introduce foundational mechanisms or demonstrate substantial conceptual advances within a paradigm. Incremental optimizations are contextualized within the evolutionary trajectory of their respective classes. The following subsections detail each class, tracing its technical evolution from foundational algorithms to recent enhancements, analyzing operational trade-offs, and discussing hybrid systems that operate at the intersections of these core paradigms.

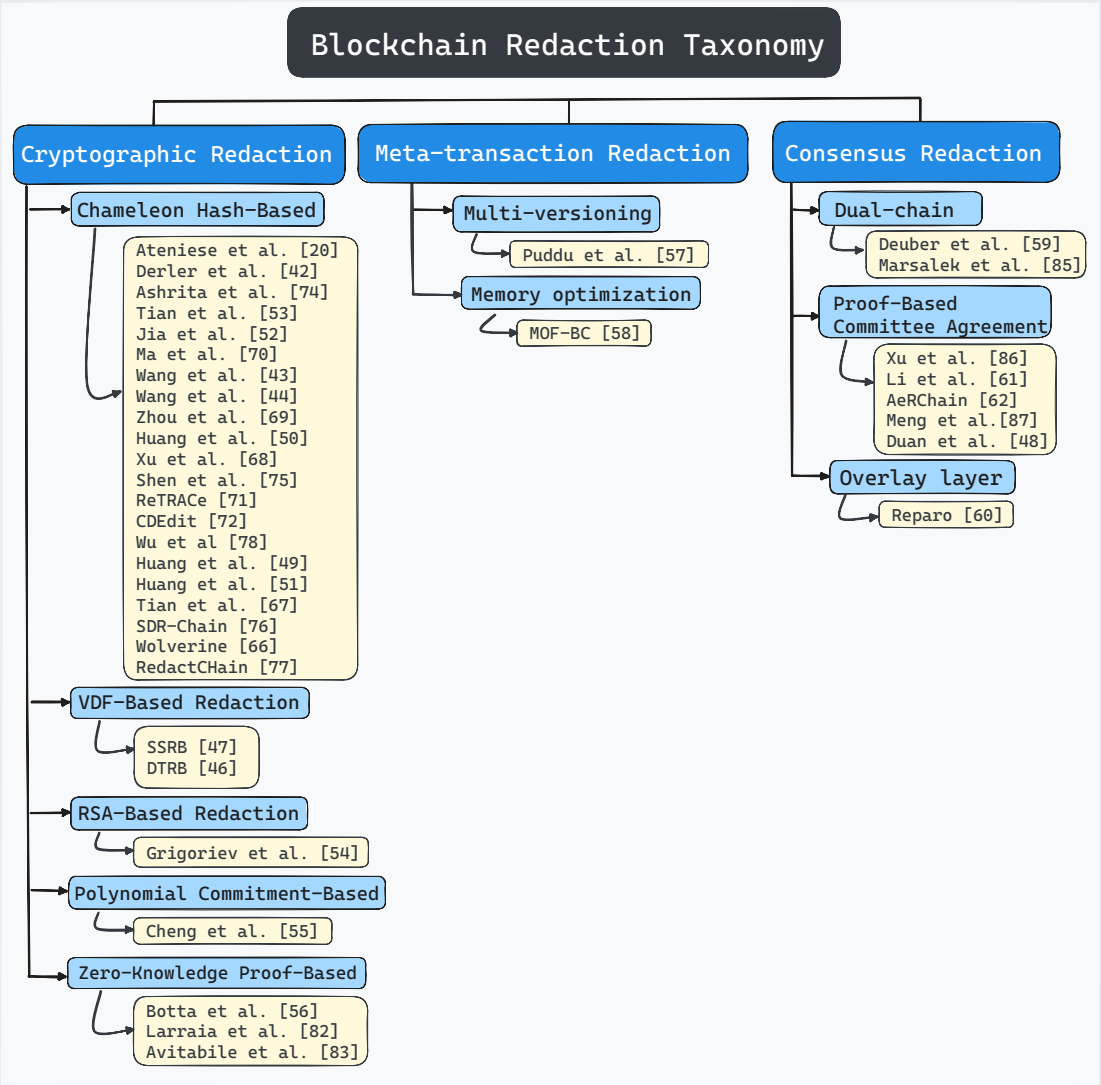


Figure 5.1: Taxonomy of state-of-the-art solutions for blockchain redaction

5.3.1 Cryptographic Redaction

Cryptographic redaction represents the most technically sophisticated category of blockchain redaction approaches, employing advanced cryptographic primitives to enable modifications while preserving the security properties of blockchain.

This approach leverages cryptographic primitives to fundamentally alter the immutability mechanisms inherent in traditional blockchains. Within this category, we identify four main techniques, including Chameleon Hash redaction, RSA based redaction, Polynomial Commitment, and Zero-Knowledge Proof redaction. Each technique offers unique trade-offs in terms of efficiency, security guarantees, and implementation complexity.

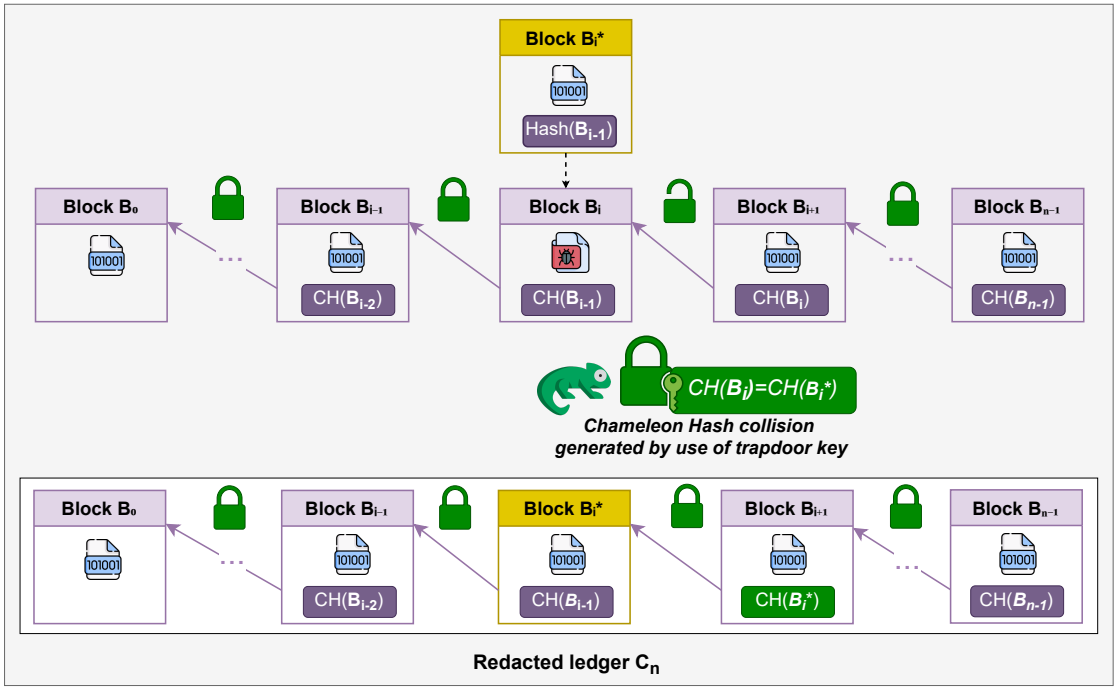


Figure 5.2: Chameleon hash redaction workflow. A block B_i^* is generated to replace the infected block B_i. A collision is computed on block B_i^* such that $CH(B_i) = CH(B_i^*)$. The block B_i^* replaces the block B_i without compromising the hash link between the adjacent blocks.

Chameleon Hash-based Redaction

Chameleon-hash based solutions operate on the key component reinforcing the immutability of blockchain: The hashing function. These solutions propose a verifiable hash as a replacement for the standard collision-resistant hash functions. The first technical proposal of a redactable blockchain was proposed by Ateniese et al.[9]. This pioneering approach, based on the concept of chameleon hashes originally introduced by Krawczyk and Rabin [112], effectively replaced conventional collision-resistant hashes with ones that allow authorized collisions under specific circumstances. It reinforced the cryptographic backbone of the blockchain while introducing a new dimension of controlled mutability. In a nutshell, chameleon hash allows to unlock the blocks hash and replace the block data, while keeping the block hash intact (See Fig.5.2).

Chameleon hash functions are a specific type of cryptographic hash functions that share similar properties as standard hash functions, such as preimage and second preimage resistance. These properties ensure that it is infeasible to determine a message that corresponds to a given hash value or discover a second message possessing the same hash as the stated message. However, chameleon hash functions have an important distinction from standard hash functions, which is the inclusion of a trapdoor key [112]. The possession of the trapdoor key enables arbitrary collision computation, which essentially breaks the collision resistance property of the hash function. The trapdoor key grants the ability to selectively rewrite certain blocks within the blockchain, while preserving the integrity of the block hash. Essentially, the only way to edit the blockchain is by using the secret trapdoor key. As a result, the loss or destruction of the trapdoor key renders the blockchain immutable, preventing any further modifications.

More concretely, considering the block format defined in 2.1, the original inner hash function G is replaced

by the chameleon hash function $Hash(hk, (s_i, x_i), r)$ with hk being the hash key, and r the randomness for the chameleon hash. This allows the holders of the trapdoor key td to edit the block data x_i without compromising the consistency of the hash link between B_i and B_{i+1}. The new expression of the hash link becomes $s_{i+1} = H(ctr_i, Hash(hk, (s_i, x_i), r_i))$. This redaction algorithm involves a change in the block structure of the blockchain (defined in expression **??**). A new field is added to the block header to include chameleon hash randomness r. The block is now in the form $B := \langle s, x, ctr, r \rangle$. The predicate of validation for a block becomes:

$$Validblock_q^D(B) := (H(ctr, Hash(hk, (s, x), r)) < D) \wedge (ctr < q) = 1 \tag{5.1}$$

A set of validators are entrusted with the trapdoor key, which grants them the authority to propose modifications. To redact a block B_i, a validator uses the trapdoor key td to compute a hash collision. The hash collision allows the validator to replace the existing data x_i within the block B_i with the new data x_i^*. The new data x_i^* and the hash randomness r_i^* replace the old data x_i and r_i, respectively, in the modified block B_i^*. Upon completion of the redaction process, the validators broadcast the updated chain as the only valid chain. All network participants are encouraged to adopt the new redacted chain and discard any alternative chains they may have.

In particular, in a permissionless setting, the redaction requires mechanisms for key distribution among blockchain miners. The validators reconstitute the trapdoor key using secret sharing schemes [167] and collectively run the redaction algorithm in a distributed manner through a secure multiparty computation (MPC). The basic idea behind the deployed secret sharing scheme [167] is to split the secret td into n shares $td_1, td_2, \ldots, td_n$ such that td can be reconstructed only if at least k out of n shares are combined, while the secret remains completely unknown with any $k-1$ or fewer td_i shares.

Beside the induced complexity of key distribution, Ateniese's scheme presents significant limitations. Redaction remains controlled by a trusted authority who exclusively holds the trapdoor, introducing a centralization risk. The security of the system depends on the honesty and competence of the validators. If the trapdoor key is compromised or the privileged validators behave dishonestly, the entire system becomes compromised.

More fundamentally, the underlying chameleon hash primitive [112] suffers from *key exposure* vulnerability. Key exposure refers to the ability of any participant to derive the trapdoor key when the first collision is computed. Such a breach enables unauthorized participants to manipulate the data without leaving any evidence.

Despite its inherent limitations, this foundational approach represents one of the most seminal contributions to the field. Most state-of-the-art solutions draw inspiration from the the original scheme of Ateniese et al., attempting to reconcile immutability with practical needs for redaction. To mitigate the risks of permanent trapdoor exposure, Camenisch et al. [36] introduced chameleon hash functions with ephemeral trapdoors (CHET). Crucially, this scheme generates a new, unique ephemeral trapdoor (etd) for each specific chameleon hash invocation (i.e., for every transaction/block), which is combined with a long-term trapdoor held by the authority to compute collisions. When a redaction is required, the transaction owner provides the specific etd associated with that hash to the authority. While the computation of a collision may technically expose the etd for that specific instance, the security model ensures this exposure is contained; since etd values are unique to each hash and cannot be reused, the secrecy of the main trapdoor remains intact, preventing unauthorized modifications to any other part of the ledger.

Many emerging solutions since adopted the CHET construction or introduced novel secure chameleon hash variants. As the field evolved, research progressively shifted from securing the trapdoor to embedding fine-grained control, accountability, and traceability into the redaction workflow. These advancements manifest differently depending on the governance model and trust assumptions of the underlying blockchain. For clarity of exposition, we first examine solutions designed for permissioned environments, then explore those in permissionless settings where decentralized coordination is required.

a) Permissioned and consortium environments.
In permissioned environments, the blockchain is maintained by a defined set of identifiable validators, and redaction control is tightly managed. In this context, the primary challenge is the enforcement of fine-grained access control and accountability among semi-trusted peers.

Derler et al. [52] introduced a Policy-based Chameleon Hash (PCH) that extends the basic chameleon-hash redaction primitive with fine-grained, attribute-driven access control. PCH specifically encrypts the ephemeral trapdoor key etd using a ciphertext-policy attribute-based encryption (CP-ABE) scheme [19], embedding a boolean access policy into the ciphertext. CP-ABE provides a more efficient access control mechanism compared to conventional public key cryptographic encryption. Only users whose attributes satisfy the access policy can decrypt etd, and thus compute a valid collision to perform a redaction.

Instead of modifying the block's hash link, users compute a chameleon hash over the transaction and insert the resulting hash into the Merkle tree. This enables redactions at the transaction level, allowing updates to individual transaction data without affecting the overall block structure or Merkle root integrity. PCH enforces fine-grained access control through attribute-based encryption, ensuring that only users satisfying the embedded policy can decrypt the ephemeral trapdoor and perform redactions. However, it does not guarantee accountability, as redactions are not cryptographically bound to the identity of the modifier once the trapdoor is revealed.

Addressing the accountability gap present in [9] and [52], Ashritha et al. [6] proposed a permissioned block-level solution incorporating mechanisms for accountability and key revocation. Their scheme introduces revocable trapdoors using non-linear secret sharing (NLSS) and distributes redaction authority among validators via MPC. While significantly enhancing security posture, their approach incurred high bandwidth and time costs and still faced key management complexities.

Table 5.3: Cryptographic-based redaction solutions.

Solution	Mechanisms	Network	Granularity	Main Features
Ateniese et al. [9]	CH, LSS, MPC, PKE	Perm./Perml.	Bl.	– Negligible redaction time – Key exposure – Tx inconsistency – No accountability/traceability – Double-spending risk
Derler et al. [52]	CHET, CP-ABE	Permissioned	Tx	– Anti double-spending – User-managed redaction – Accountability – Cost/time overhead – No public traceability

Table 5.3: Cryptographic-based redaction methods (*continued*)

Solution	Mechanisms	Network	Granularity	Main Features
Ashritha et al. [6]	CHET, NLSS, DS, MPC	Permissioned	Bl.	– Accountability – Key revocation – High bandwidth/time cost
Tian et al. [180]	CHET, CP-ABE, DS	Permissioned	Tx	– Black-box accountability – Traceability
Tian et al. [179]	CHET, ABET, DPSS	Permissionless	Tx	– Black-box accountability – Traceability
Jia et al. [101]	CHET	Perm./Perml.	Tx-field	– User-managed redaction – Key revocation – GDPR-aligned – Scalability issues
Zhou et al. [201]	ID-based CH, Homomorphic SS	Permissioned	Tx & Bl.	– Accountability/Traceability – User-managed redaction – GDPR-aligned – Dependency-aware redaction – Scalability issues
Huang et al. [96]	CHET	Permissionless	Bl.	– Hash revocation – Scalability issues
Huang et al. [97]	CH, ASCS	Permissioned	Bl.	– Accountability – Key-exposure free – Privilege misuse – High bandwidth cost – Low redaction efficiency
Huang et al. [98]	TUCH, LRRS	Permissionless	Bl.	– Detectable double-spending – Accountability/traceability – Anonymity – Computational overhead
Li et al. (Wolverine) [122]	NITCH	Permissionless	Tx-field	– Tx consistency – Accountability – Supports $< 1/2$ adversaries – Storage overhead
Ma et al. [128]	MA-PCH	Permissioned	Tx	+ User-managed redaction. + Decentralized redaction.
TPRB [184]	TPCH and CP-ABE	Permissioned	Tx	+ User-managed redaction. + Key exposure free. + Decentralized redaction. – Scalability issues.
RRB [185]	TPCH	Permissioned	Tx & Block	+ Key exposure free. + Decentralized redaction. + GDPR aligned. + Post-redaction consistency.
Xu et al. [193]	CHET, ABTT, IBS	Permissioned	Tx	– Accountability, black-box traceability – Trusted central authority – Cost/time overhead

Table 5.3: Cryptographic-based redaction methods (*continued*)

Solution	Mechanisms	Network	Granularity	Main Features
Shen et al. [168]	Double-key CH, NIZK	Permissioned	Bl.	– Full editability – State verifiability – Incentivized redaction adoption – Computational overhead
ReTRACe [147]	RCHET, RFAME	Permissioned	Tx	– Revocation Support – Accountable anonymity – Complex infrastructure
CDEdit [41]	double-key CH	Permissioned	Tx & Bl.	– Incentive-driven – Resistant to collusion
SDR-Chain [199]	CHET, DCP-ABE, LSS	Perm./Perml.	Tx	– Decentralized redaction – Backward/forward security – Dynamic participation
RedactChain [132]	CH, DKG	Permission-less	Tx	– Incentive-driven – adversarial resilience
Wu et al. [191]	Lattice-based CH	Permissioned	Tx	– Post-quantum secure – Cost and time overhead

Abbreviations. CH: Chameleon Hash. **CHET**: Chameleon Hash with Ephemeral Trapdoors. **RCHET**: Revocable CHET. **TUCH**: Time-Updatable Chameleon Hash. **NITCH**: Non-Interactive Threshold Chameleon Hash. **CP-ABE**: Ciphertext-Policy Attribute-Based Encryption. **DCP-ABE**: Decentralized CP-ABE. **ABET**: Attribute-Based Encryption with black-box Traceability. **MPC**: Multi-party Computation. **LSS**: Linear Secret Sharing. **NLSS**: Non-Linear Secret Sharing. **DPSS**: Distributed Proactive Secret Sharing. **Homomorphic SS**: Homomorphic Secret Sharing. **ASCS**: Accountable & Sanitizable Chameleon Signatures. **IBS**: Identity-Based Signature. **DS**: Digital Signature. **LRRS**: Linkable & Redactable Ring Signature. **DKG**: Distributed Key Generation. **NIZK**: Non-interactive zero-knowledge proofs.

PCHBA by Tian et al.[180] addresses accountability challenges in PCH schemes, particularly black-box key leakage where modifiers embed keys into external devices for anonymous editing. PCHBA introduced black-box accountability enabling an authority to trace redactions back to the responsible modifier even if multiple users share the same access rights. This is achieved using ephemeral trapdoors, homomorphic signatures and a traceable ABE scheme. A follow up work from [179] extends PCHBA to a decentralized version [179]. The decentralized version [179] relies on dynamic secret sharing across rotating committees to remove reliance on a trusted authority. [179] incorporates a new KP-ABE scheme with public traceability enables anyone to trace modified transactions to their origin while maintaining fine-grained access control and public accountability.

Xu et al. [193] propose a redactable blockchain design built on a Policy-based Chameleon Hash with black-box Accountability (PCHA). In PCHA, transaction owners generate a chameleon hash using CHET and associate it with an attribute-based access policy that is cryptographically bound to a novel Attribute-Based Traitor Tracing (ABTT) mechanism. The ABTT scheme allows a central authority to trace any unauthorized redaction attempt back to all users who contributed to the leaked key material, thereby enforcing black-box traceability. In addition, PCHA integrates Identity-Based Signatures (IBS) to bind each redaction operation to a verifiable identity. Modifiers who satisfy the access policy can decrypt the ephemeral trapdoor and compute a valid collision to rewrite the transaction. The use of IBS ensures that every redacted transaction is cryptographically linked to the signer's identity, enabling redaction linkability and accountability.

ReTRACe [147] composes three novel cryptographic primitives, a Revocable Chameleon Hash that extends CHET to guarantee that a revoked user cannot modify a blockchain message or trapdoor. Additionally, a revocable CP-ABE scheme is used to encrypt ephemeral trapdoors under access policies. Authorized users can anonymously post and redact messages on the blockchain, but their identities can be unmasked by legitimate oversight authorities if needed.

Ma et al. [128] propose a decentralized redactable blockchain framework to overcome the trust and centralization issues inherent in prior policy-based chameleon hash (PCH) systems. The core idea is to decentralize redaction control by distributing both the chameleon hash trapdoor and access policies across multiple independent authorities.

In Shen et al. [168], the core primitive is implemented as a double-trapdoor chameleon hash, where a signing trapdoor used during initial block creation and an editing trapdoor used exclusively for redaction. Even if one trapdoor is leaked, unauthorized redactions remain infeasible, as both are required to forge valid changes.

Shen et al. ensure global state consistency using a cryptographic accumulator that is a compact mathematical structure of valid blocks. When a redaction occurs, it triggers real-time updates to the accumulator and issues publicly verifiable proofs to reflect the changes. Membership proofs confirm active blocks belong to the current chain, while non-membership proofs invalidate redacted or historical versions.

Building on these policy-based foundations, newer constructions address residual centralization in key management. Unlike earlier redactable chains that distribute the chameleon trapdoor among a static committee using MPC protocols and secret sharing schemes, SDR-chain [199] introduces a more flexible approach by enabling dynamic participation of both redactors and attribute authorities. SDR-chain leverages a customized chameleon hash construction (DACH) that supports dynamic participation, allowing attribute authorities to join and leave without compromising functionality or security. To manage redaction keys securely SDR-chain employs Decentralized Ciphertext-Policy ABE (DCP-ABE), allowing attribute-based trapdoor generation and delegation across a dynamic set of authorities. Notably, SDR-chain ensures both backward and forward security, which guarantee that the compromise of an authority does not expose past redactions nor impact future ones.

Huang et al. have produced a sequence of redactable blockchain schemes tailored to resource-constrained industrial IoT environments starting with [97] that supports group-based redaction in consortium blockchains while maintaining accountability and per-block signature sanitization for IoT contexts. They further advanced this concept by designing a self-redactable blockchain architecture to form an intelligent, autonomous trust layer for IoT devices, enabling localized and secure redactions through collaborative decision-making among sensors [96]. Building on those foundations, their 2021 work [98] introduced a novel cryptographic primitive called the Time Updatable Chameleon Hash (TUCH) which restricts collision validity to a defined time window and avoids key sharing or coordination complexities. This temporal constraint ensures that even if a trapdoor key is compromised, it can no longer be used to generate valid collisions after expiration, which limits attack windows and mitigates key exposure risks. In this framework, the block miner generates the trapdoor key and thus holds the authority to perform redactions within the valid time window. TUCH also avoids the complexity of trapdoor key sharing among multiple users and allows spontaneous ring formation for redactions without the need for interactive or centralized coordination.Furthermore, TUCH integrates seamlessly with the Linkable and Redactable Ring Signature (LRRS) scheme, so the miner can alter transaction content without invalidating the signature. LRRS also provides strong anonymity for users while preserving linkability (links two anonymous signatures generated by the same signer) to detect double-spending. This design significantly improves security in

dynamic, trustless environments such as IoT networks.

A recent work of Wang et al. [184] proposes TPRB, a permissioned redactable blockchain that mitigates the key-exposure risk in policy-based Chameleon Hash schemes by distributing redaction authority across multiple entities. The system relies on a novel RSA-based threshold policy-based Chameleon Hash (TPCH) that eliminates reliance on a single trapdoor holder. In TPCH, the long-term trapdoor is decentralized among n authorities using Shoup's integer-based secret sharing [170], such that at least $t+1$ shares are required to enable collision generation. In parallel, the ephemeral trapdoor is protected through ciphertext-policy attribute-based encryption (CP-ABE). A modifier must both decrypt the ephemeral trapdoor under the CP-ABE policy and obtain the required threshold of shares to perform a rewrite. Then the shares are combined with Lagrange interpolation to produce a valid TCH randomness for the collision, ensuring that no single entity ever controls the full trapdoor and significantly reducing key-exposure risks.

Another work of Wang et al. [185] identifies a critical limitation in policy-based Chameleon Hash schemes, where a malicious user may encrypt an invalid trapdoor under attribute-based encryption, preventing authorized modifiers from performing legitimate edits. To address this issue, they propose a Resilient Redactable Blockchain (RRB). RRB adopts a two-level rewriting model. The first level enables standard transaction-level redactions using PCH. If that fails due to an invalid trapdoor, the system falls back to a centrally controlled block-level rewrite to force the correction. Moreover, RRB incorporates explicit version detection, embedding version identifiers that allow nodes to detect outdated states, thereby preventing reversion attacks and long-term ledger inconsistencies.

Jia et al. [101] propose a novel chameleon hash redaction scheme that supports both regulatory oversight and user self-redaction in a unified redactable blockchain framework. Their construction introduces the stateful Chameleon Hash with Revocable Subkey (sCHRS), a cryptographic primitive that distinguishes between master and subordinate trapdoor keys. The regulator holds the master key, while users generate subordinate keys that can be revoked if abused. To enhance control and reduce unintended alterations, the scheme restricts redactions to specific transaction fields such as `OP_RETURN` or coinbase outputs where arbitrary data is typically inserted, limiting flexibility by preventing edits to other data segments even when those might require correction. This field-level restriction simplifies policy enforcement but constrains the scope of permissible redactions and may leave critical errors unaddressable.

Building on the same GDPR-driven goal of combining user-initiated redactions with regulator oversight, Zhou et al. [201] propose a redactable blockchain built on an identity-based trapdoor hash without embedding explicit access policies. Instead, redaction rights are enforced implicitly through identity roles. The system supports user-initiated edits for their own transactions, and regulator-initiated deletions or dependency updates. Each user's ability to edit is cryptographically bound to a private key derived from their identity by the regulator. This ensures users can only modify their own transactions. Regulators hold the master secret to supervise the data on the chain. Regulator redaction privileges are collectively managed through a homomorphic secret-sharing scheme, which distributes the master key among multiple authorities ensuring mutual supervision between regulators. All redactions require network consensus and are publicly verifiable via on-chain metadata and consensus logs. While the approach by Zhou et al. offers one of the most controlled and regulation-aligned redaction mechanisms, this strict enforcement comes at the cost of increased system complexity, as each redaction requires an additional layer of consensus and coordinated approval, potentially impacting efficiency in practice.

Seeking more precision and consistency in data redaction, Wolverine [122] proposes a redactable blockchain that ensures transaction consistency and scalability in permissionless environments, without incurring the coordination costs of traditional MPC. The Non-Interactive Threshold Chameleon Hash (NITCH) scheme

employed by Wolverine enables collision generation without interactivity among committee members. Each member locally computes a share of the collision based on their private trapdoor fragment, and the shares can be later aggregated by any participant to reconstruct the full collision. In Wolverine [122], a regulatory committee elected via Sybil-resistant mechanisms collectively holds shares of a trapdoor key for a NITCH. Crucially, NITCH enables: (i) Each committee member holds a secret share (sk_i) of the master trapdoor. (ii) Collisions can only be generated with $t+1$ shares ($t < n/2$), preventing minority control. Any user in the network can initiate a redaction request by embedding it into a transaction. The committee evaluates the redaction request according to redaction policy ensuring that the modification targets only admissible fields and the transaction consistency is preserved. Upon approval, each committee member independently computes a partial collision share and broadcasts it as an on-chain vote transaction. Miners aggregate valid votes into blocks, and once a majority of committee votes approve the redaction request, shares are combined to create a valid hash collision and execute redaction enabling data redaction.

Importantly, Wolverine enforces transaction consistency throughout this process. It models dependencies between transactions formally and prevents any redaction that would violate semantic validity or break links in subsequent blocks. If a redaction affects transactions with dependencies, it is either rejected or requires coordinated cascading redactions to preserve integrity. Finally, to ensure long-term security and mitigate trapdoor compromise, Wolverine periodically rotates the redaction committee using randomness from Decentralized Random Beacons (DRBs) [72] derived from NITCH ensuring that redaction authority is periodically reassigned.

Unlike prior schemes that expose redaction capabilities to policy-satisfying users without usage constraints, CDEdit [41] introduces a token-governed redactable blockchain framework that emphasizes controllable privilege delegation and multi-level editing. CDEdit introduces a Privileged Token Service (PTS) to issue edit tokens that limit both the granularity (Tx-level vs. Block-level) and number of permitted edits (n-times per token) per modifier. It combines CHET with a fast and low overhead attribute-based encryption scheme (FAME) [2], to encrypt the ephemeral trapdoor under fine-grained policies.

RedactChain [132] presents a hybrid redaction framework that combines threshold chameleon hash functions (TCHFs) with a jury-based voting mechanism to enable controlled mutability in permissionless blockchains. Redaction authority is decentralized through a periodically rotated jury, whose members collaboratively generate chameleon hash collisions using a distributed key generation (DKG) protocol. Redactions are permitted only within a defined window during the jury's tenure, ensuring time-bounded privileges and limiting the risk of misuse. To ensure transparency, the system maintains an immutable redaction log recording every approved modification. RedactChain also differentiates its content handling strategy through removal for non-spendable outputs (e.g., `OP\_RETURN`) and obfuscation for spendable outputs to preserve transaction validity. Furthermore, The design provides formal adversarial resilience, tolerating attackers with up to 25% of mining power. Despite its robustness, the approach remains tightly coupled to the Bitcoin transaction model and introduces coordination overhead due to jury management and voting synchronization.

While prior solutions ignored quantum threats, Wu et al. [191] present two single-trapdoor, key-exposure-free chameleon hash schemes relying on lattice-based assumptions marking a quantum-secure design. Wu et al. address two critical gaps in redactable blockchains, existing chameleon hashes rely on discrete-log/factoring assumptions, which can be broken by quantum attacks, and the problem of key exposure. In the proposed design, each transaction or block is associated with a chameleon hash value, generated using a lattice-based one-way function. When a redaction is needed, any party possessing the public trapdoor can compute a collision.

The first construction assumes that the trapdoor is made fully public. In this scheme, anyone can compute collisions, but redactions are controlled externally through a voting mechanism. Wu et al. [191] also propose a distributed variant where the trapdoor is shared among multiple parties using secret sharing or lattice-based threshold trapdoor generation via distributed GPV sampling. In this version, redactions are only possible if a threshold number of parties cooperate to reconstruct a valid short preimage, enabling committee-based redaction.

VDF-based Redaction

Most cryptographic redactable blockchain schemes rely on chameleon hashes. While these approaches allow modifications without breaking hash links, they generally fail to enforce global consistency across all nodes, as they rely on voluntary adoption. Wang et al. first proposed SSRB [186], a redactable blockchain for permissioned settings that addresses these consistency issues using Verifiable delay functions (VDFs). VDF–based redaction introduces a different paradigm by treating time as a verifiable resource. A VDF requires sequential computation over a specified time parameter, and produces a proof that can be efficiently verified by any node. Under normal operation, blocks are generated with a fixed time difficulty, ensuring sequential computation. Redaction in SSRB is realized through chain reconstruction. When a redaction is authorized, a trusted authority holding the VDF trapdoor can quickly recompute the VDF outputs for the modified block. All subsequent unmodified blocks must be recomputed to maintain hash linking, forming a reconstructed chain. The reconstructed chain has the same length as the original chain but a larger cumulative VDF time value, since redacted blocks are assigned a higher effective time parameter. SSRB enforces synchronized redaction through a longest-time rule; nodes must adopt the reconstructed chain to remain on the main chain. Nodes that refuse to synchronize cannot validate or extend future blocks, since blocks extending stale history reference a hash pointer different from that of the reconstructed chain head. However, SSRB relies on a centrally held VDF trapdoor, introducing a single point of trust.

Building on SSRB, the authors later introduced DTRB [183], which replaces the single trapdoor with a threshold-shared trapdoor VDF (DTVDF). In DTRB, the secret VDF trapdoor is split into shares distributed among multiple participants. Redaction requires collaboration from a threshold of nodes to reconstruct the trapdoor and accelerate the VDF computation thus it ensures that no single entity can unilaterally modify the chain. Redaction in DTRB proceeds as follows. After a redaction proposal is approved, the leader node identifies the earliest block to be modified and initiates collaborative chain reconstruction. For each affected block, participating nodes independently compute DTVDF result shares along with proofs of correctness. The leader then combines a sufficient set of shares via Lagrange interpolation to reconstruct the accelerated VDF output and rebuild the updated chain segment. Once the modified chain segment is reconstructed, DTRB further improves efficiency through aggregate verification. Instead of verifying each redacted block's VDF proof individually, DTRB introduces an aggregated proof for the entire reconstructed chain at once and provides accountability by embedding the modifier's signature into aggregated proofs. Overall, DTRB maintains the consistency guarantees of SSRB while restoring decentralization.

RSA-based Redaction

Grigoriev et al. [82] a redactable blockchain method specifically designed for private blockchains, based on the computational hardness of the RSA problem. In an RSA-based redactable blockchain, each block contains three core components: a permanent prefix (ensuring immutability of structural links), modifiable

content, and a redactable suffix (enabling authorized edits). The permanent prefix P_i is a fixed value derived from the previous block's hash, ensuring unbreakable links between blocks $P_i = H(P_{i-1}, C_{i-1})$ with H is a collision-resistant hash function. The modifiable content C_i represents the data that authorized entities can edit. The redactable suffix X_i is a cryptographic value computed using RSA operations, enabling secure and verifiable updates to the modifiable content.

Blocks are cryptographically linked through a hash function applied to the prefix and content, generating a hash value. The suffix is calculated by converting h_i into an integer d_i modulo n (where $n = pq$ is the product of two large primes p and q). In the RSA-based redactable blockchain, the central authority possesses the RSA private key, which consists of two critical secret components: (i) Prime factors p and q of the modulus $n = pq$. (ii) The function $\phi(n) = (p-1)(q-1)$. These elements enable the authority to compute modular inverses and generate valid suffixes during redactions.

Technically, the RSA private key is used to derive e_i a modular inverse of d_i such as $e_i \equiv (d_i) \bmod \phi(n)$. The suffix X_i is then calculated such as $X_i \equiv P_{i+1}^{e} \bmod n$ where P_{i+1} is the prefix of the next block. This enforces that:

$$X_i^{d_i} \equiv P_{i+1} \bmod n \tag{5.2}$$

This congruence ensures that only valid suffixes (computed with d) can satisfy the relationship to P_{i+1}.

To redact a block, a trusted authority replaces the original content C_i of a block B_i with new content C_i^*, recomputes the hash $h_i^* = H(P_i, C_i^*)$, derives a new integer d_i^*, find e_i^*, the modular inverse of d_i^* and adjusts the suffix $X_i^* \equiv P_{i+1}^{e_i^*} \bmod n$ using the RSA private key. Critically, the prefix P_{i+1} remains unchanged, preserving the link to subsequent blocks.

$$(X_i^*)^{d_i^*} \equiv P_{i+1} \bmod n \tag{5.3}$$

The security of RSA-derived method stems from the fact that recovering X_i^* from P_{i+1} and d_i^* without knowing p, q, or d is computationally infeasible (equivalent to solving the RSA problem). Also, Since P_i depends on prior blocks, tampering propagates detectably. Hence, if an adversary tries to revert to C_i and X_i, they must modify the block's current state, which would break the link: $(X_i)^{d_i^*} \not\equiv P_{i+1} \bmod n$ since $d_i^* \neq d_i$. However, if a malicious redactor maintains a private log of prior block content C_i and suffix X_i, they can revert the blockchain to any logged state by reapplying the redaction process [57].

Polynomial Commitment-Based Redaction

The polynomial-based blockchain introduces a novel structure where each block is cryptographically linked through polynomials rather than hashes [42].

For each field in the block (header or data), a hash-cut function `HASHCUT` is applied to produce a unique 2D coordinate (x_i, y_i). This function hashes the data value with other metadata and padding, then extracts numeric values from the hash output $x_i = \texttt{HASHCUT}(H(tx_i \| \text{metadata} \| \text{padding}))$ and y_i is chosen from a specific modification difficulty range. For the header specifically, the coordinate (x_0, y_0) is derived by applying `HASHCUT` to the hash of the previous block's polynomial $L_{i-1}(x)$, metadata and padding $(x_0, y_0) = \texttt{HASHCUT}(H(L_{i-1}(x) \| \text{metadata} \| \text{padding}))$. By mapping each transaction to its corresponding coordinate, we get a set of labeled points $\{(x_0, y_0), (x_1, y_1),$
$\ldots, (x_k, y_k)\}$. Using these points, a unique interpolation polynomial $L_i(x)$ is constructed using Lagrange interpolation. The resulting polynomial $L_i(x)$ is a cryptographic fingerprint of block B_i, replacing the block

hash. The next block B_{i+1} includes in its header a point(x, y) such that $L_i(x) = y$. This process mirrors how traditional blockchains reference the hash of the previous block. Here, blocks are linked through polynomial consistency; each new block includes a point from the polynomial defined by the previous block, ensuring that any tampering breaks the chain. Modifications require recalculating these coordinates to satisfy the polynomial relationship. The system allows users to define a modification difficulty that refers to a range or set of valid ordinate values ($y \in [a, b]$) that new coordinates must satisfy during redaction.

Redaction in this scheme is achieved by carefully modifying a field (data or header) to a new value tx^*. The system then must re-map this new value to a new coordinate (x^*, y^*) via a hash-cut operation, such that x^* is unique and does not collide with existing coordinates. Most importantly, the coordinate (x^*, y^*) must still lies on the existing polynomial curve of $L(x)$ such that $L(x^*) = y^*$. This ensures the redacted block is still consistent with the polynomial structure of the chain. To avoid trivial tampering, the system implements a "modification difficulty" constraint that restricts valid y^* values to a specific range $[a, b]$.This mechanism acts like a PoW puzzle, requiring computational effort to find valid replacement coordinates that maintain polynomial consistency across the blockchain while satisfying the difficulty constraint.

The ability to set modification difficulty allows the same blockchain structure to serve applications with varied security requirements. Furthermore, the difficulty parameters allow users to enforce a high degree of control over their data. Transactions can either be set immutable by configuring difficulty parameters to require infeasible computation or support potential redaction with lower settings. Nonetheless, such a mechanism requires knowledgeable users of the impact of this difficulty parameters to choose the value that properly balances security with redaction rights.

The polynomial implementation necessitates larger storage requirements in both headers (storing polynomials, primes, and coordinates) and body elements (storing coordinates for each transaction record). This implies that The polynomial structure sacrifices storage efficiency for operational flexibility, making it more suitable for applications where data updates outweigh storage costs (e.g., healthcare records, supply chain tracking).

Zero-Knowledge Proof-based Redaction

In [23], the authors propose a framework that enables Bitcoin nodes to redact specific non-financial data from transactions using zero-knowledge proofs. The technique targets auxiliary data such as information embedded in coinbase or `OP_RETURN` outputs, without affecting the financial semantics or exposing the system to double-spending vulnerabilities. Technically, the mechanism employs zero-knowledge proofs to prove partial knowledge of hash preimages corresponding to the redacted date, without revealing the deleted data.

A redactor N identifies a transaction tx containing data to delete. The redactor creates a modified transaction tx^* by replacing sensitive data with zeros. Also, it retains the original hash $h = Hash(t)$, which serves as a cryptographic commitment to the unaltered data.

The redactor N replaces tx with tx^* in its local blockchain copy. Fundamentally, the Merkle tree root in the block header remains unchanged. Instead of recomputing the root after modifying tx, N keeps the original hash h in the Merkle leaf for tx^*. Additionally, N generates a non-interactive zero-knowledge proof (NIZK) π corresponding to this redaction for verification. The proof π proves that there exists some original data that, when inserted into tx^*, produces a transaction tx^{**} whose hash matches the original hash h. In other words proof ensures that tx^* is a legitimate redaction of tx, even though the original data

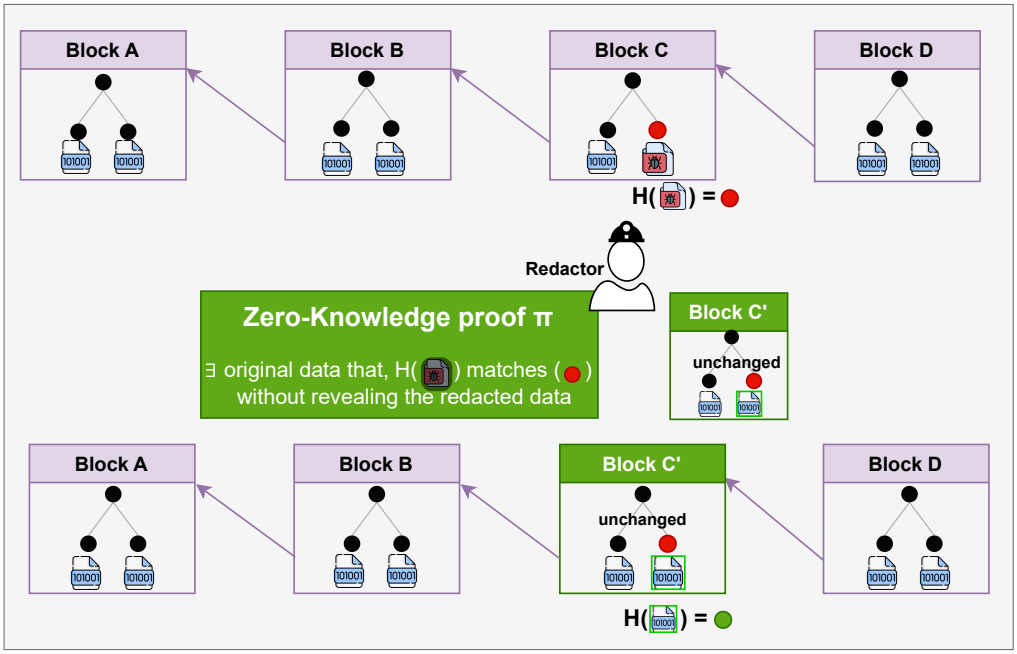


Figure 5.3: Zero-knowledge proof redaction. After redacting block C, the old hash is kept on the merkle tree. The redactor generates a zero-knowledge proof π that proves the existence of original data such that the hash of the redacted data matches the original hash, without revealing what was redacted.

is hidden. N replaces tx with tx^* in its local copy of block B and deletes the original redacted data. Figure 5.3 illustrates the process of redaction.

When another node P requests block B, the redactor N sends the modified transactions tx^* with the original hashes h and the NIZK proofs π. When P receives a redacted block B, it initiates a verification procedure to ensure the validity and integrity of the redacted data. First, P verifies the zero-knowledge proof π associated with the redacted transaction tx^*. This verification confirms that the original hash h corresponds to a valid transaction and that the redacted bytes are confined to permissible positions. Next, P reconstructs the Merkle tree in such a way, for unmodified transactions, it uses their actual transaction hashes as Merkle leaves and for redacted transactions, it substitutes the original hash h (extracted from the NIZK proof statement) as the corresponding leaf. Finally, P computes the Merkle root from these leaves and checks its consistency with the root included in the header of block B. If the computed root matches the header root, the redacted block B is deemed valid.

The proposed data redaction solution offers a balance between blockchain immutability and regulatory compliance by enabling nodes to delete specific data while preserving the consistency of the chain. A distinguishing strength of this approach is its post-quantum security, which ensures that redacted content remains protected against future cryptographic vulnerabilities. Additionally, the solution supports local redaction autonomy, allowing individual nodes to act without consensus, and cryptographic transparency via zero-knowledge proofs that validate deletions without revealing sensitive data. The solution maintains public verifiability, ensuring that all transactions remain consistent and independently verifiable after redaction, which preserves the foundational trust mechanism of distributed ledgers.

While the solution effectively addresses regulatory tensions and ethical considerations by providing a mechanism to remove illicit content, it fails to align with the storage constraints inherent to blockchain systems. Specifically, generating and storing cryptographic proofs introduces computational and storage overhead. Additionally, the lack of global coordination in the redaction process risks creating fragmented versions of the blockchain across different nodes.

Another improvement by Larraia et al. [120] offering formal security guarantees and technical improvements over Botta's solution [23]. The blockchain is partitioned into equivalence classes based on system-wide policies. Two blockchains belong to the same class if they agree on critical validation data such as the UTXO state, but they may differ in non-essential, redactable data. Nodes can store different versions of the blockchain as long as they belong to the same class.

Building on Ateniese et al.'s redaction framework for blockchains [9], the authors of [10] extend the concept to address state consistency issues introduced by redactions in smart-contract-enabled, permissioned blockchains. Their work specifically targets the challenge of updating transactions without compromising the correctness of smart contract execution or the integrity of the global state. The proposed approach leverages the zero-knowledge-based redaction techniques from [23] to construct proofs-of-consistency that validate post-redaction state correctness.

The redaction process begins with the identification of the transaction to be modified. Using chameleon hash functions, the redacting authority constructs a substitute block where the targeted transaction tx is altered or removed. The system then identifies all smart contract states and subsequent transactions that depend on tx. For each affected transaction, the redacting entity generates SNARK proofs attesting that state transitions remain valid despite the redaction. These proofs certify that (i) the original transaction tx produced a valid intermediate state st', (ii) subsequent transactions $\tilde{tx}$ validly transitioned the state from a state st' to a final state st'', and (iii) the chameleon hash of the modified block remains consistent with the original block hash recorded on-chain.

This framework enforces modifications to directly affected transactions, while all dependent transactions are subsequently validated through their attached proofs. However, this scheme is most effective for redactions that involve deletions or neutral updates; any change to transaction semantics risks violating smart contract logic and causing inconsistencies. Additionally, the method presumes a trusted governance model to manage trapdoors and proof generation, which limits its applicability in permissionless blockchain environments.

Summary. Cryptographic approaches achieve redaction by fundamentally augmenting or replacing the immutable hash linkage of conventional blockchains with specialized primitives. Most prominently chameleon hashes, but also RSA-based commitments, polynomial commitments, and zero-knowledge proof schemes. These techniques embed a controlled trapdoor key (often distributed via secret-sharing or attribute-based encryption) that permits authorized collision-finding and block or transaction rewrites without breaking the inter-block hash chain. Variants address key-exposure (ephemeral or time-limited trapdoors), accountability (traceable ABE, linkable signatures), and quantum resistance (lattice-based constructions). While offering fine-grained redaction and strong cryptographic guarantees, they incur nontrivial overhead in key management, proof generation, and validation complexity, and often require permissioned or semi-permissioned governance models.

Cryptographic redaction approaches enable direct modification of blockchain content by altering its underlying immutability primitives. These solutions provide strong cryptographic guarantees. They preserve the verifiability of redacted chains, ensure auditability, and allow fine-grained redaction control through trapdoor

mechanisms or policy-driven encryption. In particular, they support transaction- or field-level mutability while maintaining the structural integrity of the chain, making them well-suited for sensitive applications in regulated or high-assurance environments. However, these benefits come at a cost. The implementation of such techniques often introduces substantial complexity in cryptographic design, key management, and

Table 5.4: Cryptographic-based redaction solutions (continued).

Solutions	Mechanisms	Network	Granularity	Main Features
SSRB [186]	VDF	Perm.	Block	+ Strong synchronization. + Support dynamic nodes. – Centralized redaction. – Computational overhead.
DTRB [183]	DTVDF & NIZK	Perm.	Block	+ Strong synchronization. + Accountability + Decentralized redaction. – Secret-sharing overhead.
Grigoriev et al. [82]	RSA	Perm.	Block	– Strong cryptographic security. – Anti double-spending. – Centralized redaction authority. – No accountability or traceability. – Key Management problems.
Cheng et al. [42]	Lagrange polynomials	Perm.	Block	– Adjustable Security. – User-managed redaction. – Not compatible with competitive consensus (e.g., PoW). – No accountability or traceability. – Storage Overhead. – Computational overhead for high-degree polynomials.
Botta et al. [23]	zk-SNARK	Perml.	Tx-field	– Adjustable Security. – Post-quantum secure. – Data Consistency. – Redaction accountability and traceability. – Bitcoin-specific. – Limited fields for redaction. – Storage Overhead.
Larraia et al. [120]	zk-SNARK	Perml.	Tx-field	– Enhanced Security. – Post-quantum secure. – Data consistency. – Redaction accountability and traceability. – Generalized to PoW. – Limited fields for redaction. – Storage overhead.
Avitabile et al. [10]	CH & zk-SNARK	Perml.	Block	– Chain Consistency. – Redaction accountability and traceability. – Public Verifiability. – Centralized Governance. – Key Management. – Storage Overhead.

Abbreviations. **VDF:** Verifiable Delay Function. **DTVDF:** Decentralized trapdoor VDF. **RSA:** Rivest-Shamir-Adleman. **Zk-SNARK:** Zero-knowledge succinct non-interactive arguments of knowledge.

coordination protocols, especially in decentralized settings. Some schemes risk centralization if redaction authority is poorly distributed, while others suffer from scalability limitations due to computational or storage overhead. These challenges have motivated the exploration of alternative paradigms that prioritize simplicity, operational efficiency, and policy-driven control. Notably, meta-transaction-based approaches shift the redaction logic from cryptographic mutation to logical reinterpretation, achieving redaction through transaction versioning and access control without modifying the underlying block structure.

5.3.2 Meta-transaction based Redaction

In contrast to cryptographic solutions that modify block contents through specialized cryptographic schemes, meta-transaction-based redaction operates through semantic override. This approach preserves the strict data immutability of the ledger while enabling logical modifications through controlled state interpretation. Redaction is achieved by appending explicit corrective transactions that reinterpret specific past entries without physically altering them. Consequently, the modification occurs strictly at the state-interpretation level; the original data remains intact, but the canonical view of the state is updated to reflect the redaction. In the following, we examine representative techniques in this class and analyze their mechanisms, guarantees, and limitations.

A foundational system in this class is μchain proposed by Puddu et al. [155]. μchain stores multiple versions of data, but designates only one as the active, valid state. Redactions are executed through special mutation transactions that alter which version is considered active. The outdated version persists on the chain but is cryptographically obscured, as validators stop distributing the keys needed to access it.

Formally, in μchain, each transaction is defined as a triplet (T, a, P), where T is a transaction set containing multiple encrypted versions τ_i, a is the index of the currently active (decrypted) version, and P is a mutation policy specifying who can issue redactions or extensions, and under what conditions. A mutant transaction $(Ref_{\hat{T}}, a')$ activates a different transaction version by referencing an existing transaction set and updating the active index. To add new alternatives, an extension transaction $(Ref_{\hat{T}}, T_x)$ appends new versions $T_x = \tau_{k+1}, \ldots, \tau_m$ to the original set. Special *nope transactions* also allow marking a transaction as logically deleted.

When a mutable transaction T is sent, the sender specifies the active transaction index a in the transaction set τ and provides a mutation policy P for the mutable transaction indicating how it may be modified (Figure 5.4 *step 1*). Upon receiving transactions, miners add them to the blockchain and make only the active transaction available by decrypting it (Figure 5.4 *step 2*). Subsequently, a mutator M, determined by the mutation policy of the submitted transaction T, submits a mutant transaction T'. The mutant transaction T' contains a reference to the transaction to be modified Ref_T and indicates a new active transaction version $a' \neq a$ from the transaction set (Figure 5.4 *step 3*). Once the new version is validated by the set of validators, it is finally added to the next block, making the decryption keys of previous versions unavailable (Figure 5.4 *step 4*). Similarly, authorized users can extend the transmitted transactions (w.r.t., the mutation policy) by issuing an extending transaction.

The μchain is presented in both permissioned networks, where designated validators manage decryption keys for active transactions, and permissionless networks, which require a secret sharing scheme for key distribution. To handle dynamic validator groups and network churn, μchain integrates a dynamic proactive secret sharing (DPSS) scheme [15] alongside Shamir's secret sharing scheme [167].

The μchain offers a high level of transaction consistency post-redaction. The system processes the history of

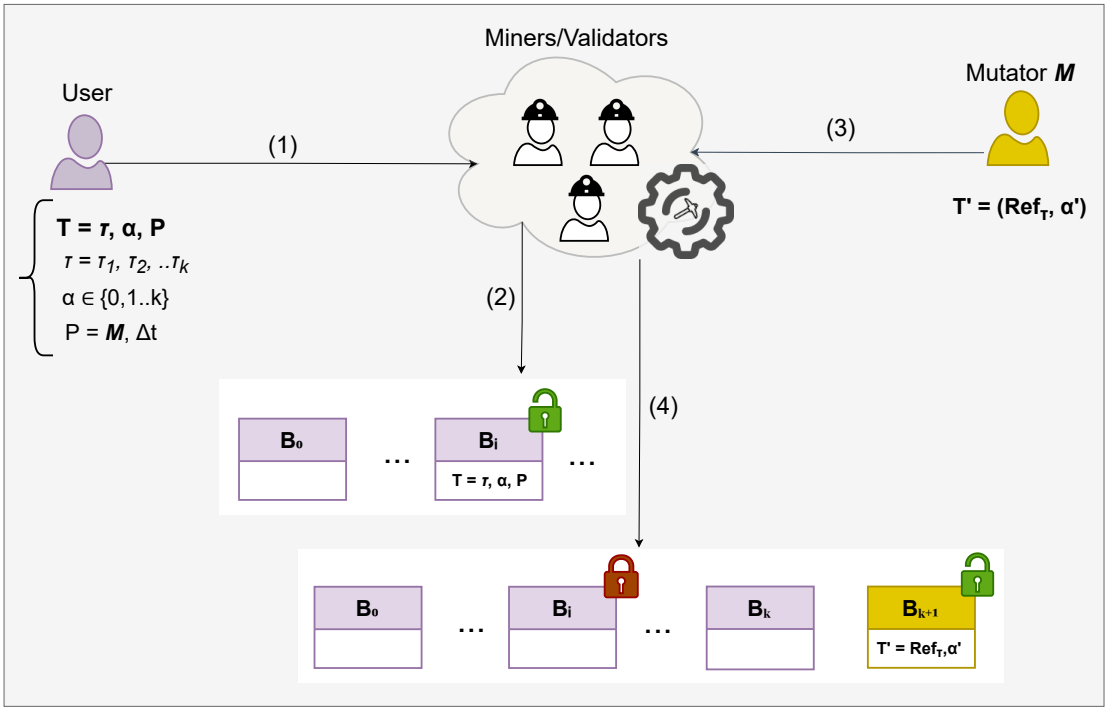


Figure 5.4: μchain redaction flow. (1) A user submits the mutable transaction $T = (\tau, a, P)$. (2) Miners validate the transactions and serve the decryption keys to the recipients. (3) A valid mutator M submits a mutant transaction T' to mutate T. (4) The miners verify and validate the mutation and add it to the chain, and stops serving the decryption keys for the old data.

a mutable transaction set and their dependencies; when a transaction is modified, the system automatically triggers the necessary updates of dependent transactions to maintain validity and prevent double-spending, ensuring transaction consistency.

More pronoucedly, μchain does not directly edit the block data with the old transactions, therefore no modifications to the block hash are required. Modified transactions are generated and recorded in next blocks. As a result, obsolete transactions remain on-chain in encrypted form, hidden but not removed. While this approach maintains blockchain integrity, it incurs storage overhead due to persisting encrypted inactive transactions indefinitely. Furthermore, reliance on multiple encryption keys necessitates complex secure key management (distribution, rotation, revocation), impacting scalability and performance through added cryptographic operations.

Another work of Dorri et al. [56] presents MOF-BC a blockchain redaction framework designed to address storage scalability and privacy challenges in IoT networks. MOF-BC [56] permits users to delete or summarize their own transactions, enabling a form of *"right to be forgotten"* while preserving overall consistency.

To enable redaction without compromising privacy or requiring long-term key storage, MOF-BC introduces the Generator Verifier (GV). The GV is a unique identifier for each transaction, created by signing the hash of the previous transaction identifier $P.TxID$ which establishes the link between successive transactions created by the same node. And a Generator Verifier Secret (GVS) corresponding to some user-held secret. All GVs are linked to a single public key ($GV\text{-}PK^{+}$), so users only need to maintain one private key. In

practice, when a user wants to remove a transaction they created, they issue a special *redacting transaction*. The *redacting transaction* records the hash of the deleted transaction and the hash of the previous transaction to ensure the transaction consistency. The *redacting transaction* also contains a proof of ownership created by hashing a secret known only to the user GVS and the previous transaction ID *P.TxID*, as well as their public key ($GV\text{-}PK^{+}$).

The network verifies this *redacting transaction* by checking that the revealed GVS and *P.TxID* matches the stored GV in the original transaction, and that the signature is valid, proving that the user knows the corresponding PK^{-} to the $GV\text{-}PK^{+}$. Once validated, *Redacting transactions* are queued and processed in batches during a designated cleaning period to reduce overhead compared to per-transaction processing.

MOF-BC ensures that block integrity remains valid after data removal. In effect, block hashes are computed over transaction hashes and not their content, which allows removal without breaking hash consistency. Thus, when a transaction content is removed, its original hash remains in the block header, so the chain of block hashes is valid.

MOF-BC implements a dynamic fee structure based on the storage duration and size of transactions. Users are incentivized to optimize memory usage through rewards, encouraging behaviors like data summarization and timely deletion. The framework supports flexible Memory Optimization Modes (MOMs), such as temporary storage and summarization, reducing storage overhead by up to 25% compared to conventional blockchains [56]. On the other hand, While MOM flags and GV fields add flexibility, they also enlarge transaction size and impose extra processing overhead on miners and agents to detect and execute redaction meta-transactions. Similarly, the batch processing delay (Cleaning Period) means redactions are not immediate, which may not suit real-time applications. Furthermore, MOF-BC's model confines redaction authority strictly to transaction owners, which limits redaction scope in some legal or regulatory scenarios.

Summary. Meta-transaction-based approaches shift the redaction paradigm from cryptographic rewriting to semantic reinterpretation. Instead of modifying existing data, they introduce auxiliary transactions that overwrite, mask, or extend historical records, while preserving the original data on-chain in encrypted or inactive form. These methods offer strong auditability, lightweight mutability, and low protocol intrusion, making them attractive for blockchains where historical data should be hidden but not deleted (e.g., forensics, compliance reversibility). They are generally easier to deploy in permissioned or resource-constrained environments. However, meta-transaction methods retain the original data, making them vulnerable to privacy leaks and storage inefficiencies. Their reliance on encryption and key secrecy also introduces key management challenges.

5.3.3 Voting-based Redaction

Voting-based redaction represents a distinct class of approaches that address the redaction problem by operating directly on the second pillar of blockchain immutability: consensus. Instead of entrusting redaction authority to individual validators through key-based mechanisms, these systems introduce collaborative agreement through explicit voting protocols. Participants evaluate redaction proposals and collectively approve or reject them based on predefined criteria such as quorum thresholds, policy compliance, or weighted stake. This section examines notable solutions that take advantage of the voting-based redaction approach, highlighting their benefits and implications in mutable blockchain networks.

Deuber et al. [53] proposed the first voting-based algorithm for block redaction adapted for permissionless

settings regardless of the deployed consensus protocols. The proposal leverages dual link to implement data modification so that two hash links are retained between adjacent blocks. The initial link connects the blocks as they were initially created and mined, while the correction link refers to the modified new data at the block granularity, as shown in Figure 5.5.

Users issue redaction requests to update the content of a block in the chain by proposing a candidate block with new data and submitting it for validation. A set of validators check the compliance of the proposed block with the redaction policy. The redaction policy essentially outlines the level of agreement under which data can be redacted. The validators initiate an on-chain voting process to vote for valid candidate blocks within the voting window specified in the redaction policy. The policy sets the vote threshold and specifies the duration of the voting process, which is often measured in terms of the number of blocks mined after the redaction request. The voting threshold indicates the number of votes needed from the miners in order for the redaction to proceed. If the redaction proposal receives a majority of votes that surpasses the specified voting threshold, it is considered approved. In this case, the old block containing the data to be redacted is replaced by the updated version reflecting the approved changes. This updated block is then broadcasted to the network, ensuring that all participants have access to the modified data.

Technically, this model adds a field y_i to the block B_i to store the initial state of the original block data, $B_i = \langle s_i, x_i, ctr_i, y_i \rangle$ such that $s_i = H(ctr_{i-1}, G(s_{i-1}, x_{i-1}), y_{i-1})$ is the hash of the previous block and $y_i = G(s_i, x_i)$ contains the initial hash of the block B_i upon its creation and addition to the blockchain.

Any user can propose an edit for a block B_i by creating a new candidate block $B_i^* = \langle s_i, x_i^*, ctr_i, y_i \rangle$. The candidate block B_i^* is then submitted to the validators for verification. First, it is verified against the redaction policy. If valid, an on-chain voting mechanism is launched; the nodes vote for the candidate block by adding its hash $H(B_i^*)$ to the data x of the next block as such $x \| H(B_i^*)$.

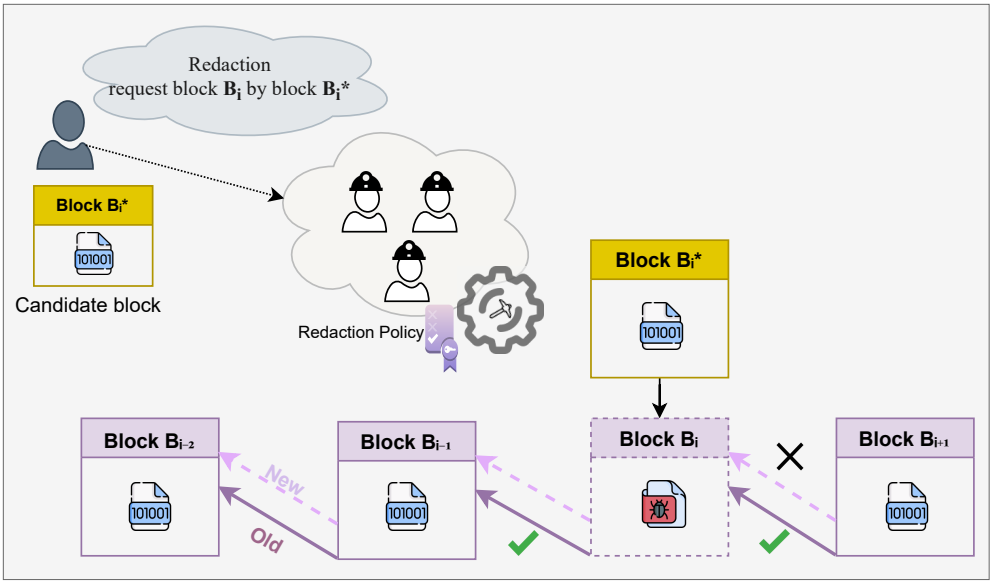
If the number of votes exceeds the threshold ρ specified in the policy, the network can proceed with the redaction. Upon redaction i.e., The block data x_i is modified to the new data x_i^*, the hash link between B_i and B_{i+1} is compromised. However, since the original copy of the block hash kept in y_i is intact, the original hash will still hold for verification.

When such a situation is encountered, the validators employ several steps to ensure the validity of the redaction. First, validators check the validity of the data in the proposed block. Second, they verify if the block has solved the proof of work (w.r.t. old links). Third, if the new link introduced by the edit is broken, validators further validate the integrity of the old links in the blockchain. Finally, validators check whether the proposed edit complies with the predefined policy to proceed in the redaction process.

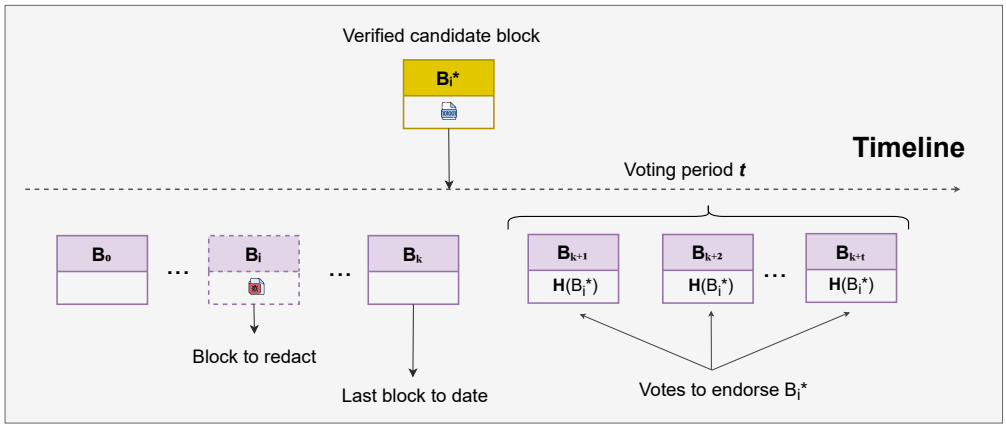
If these aforementioned conditions are met, it confirms the authentication and validity of the proposed block edits. In such scenario, the modifications are considered authorized and accepted within the blockchain system. However, if any of the conditions are not satisfied, this implies that the block has been tampered with without proper authorization.

In summary, the redaction method presented in [53] offers a reliable and accountable way to redact blocks in permissionless blockchain systems. However, the voting period required for approving valid redactions must be sufficiently long to ensure honest redaction approval, leading to poor scalability and low efficiency. Moreover, the alteration of the block structure renders this redaction method incompatible with standard blockchain models.

Building on Deuber's framework, Marsalek et al. [131] propose a blockchain architecture with two



(a) Redaction request handling. When a user submits a redaction request, miners first verify the candidate block against the redaction policy. Then, they check the integrity of the updated and original links with adjacent blocks.



(b) Voting phase. After candidate block verification, a voting period t begins, where miners vote for the block by adding its hash to the blocks they mine afterward.

Figure 5.5: Voting-based redaction workflow.

coordinated chains, a *Standard Chain* preserving the canonical transaction history, and a *Redaction Chain* maintaining tamper-proof logs of authorized data modifications. Similar to the governance model in Deuber's framework, redactions in Marsalek et al.'s system are incorporated only after a majority vote by authorized participants. The redaction itself is recorded as a redaction block appended to the Redaction Chain, ensuring that the history of modifications is publicly visible and auditable. A key contribution of Marsalek et al. is the support for arbitrary data redactions, extending beyond `OP_RETURN` fields or metadata.

Another solution by Xu et al. [192] proposes redaction of data in permissionless Proof-of-Stake (PoS) blockchain that achieves fast redaction confirmation. The key novelty lies in leveraging the PoS slot

structure for both block production and redaction voting, allowing edits to be proposed, voted on, and confirmed within a single block in synchronous networks. A committee of validators, selected using Verifiable Random Functions (VRFs) proportionally to their stake, is responsible to vote on redaction proposals. Redaction proposals are included in the block produced by the slot leader, a validator randomly chosen in proportion to their stake within an assigned time slot. The slot leader aggregates validator signatures into a single multi-signature (`msig`), proving that multiple signers jointly signed the redaction vote. Both the `msig` and VRF proofs are recorded on-chain, and once the quorum threshold is reached, the redaction becomes final.

Li et al. [124] propose a redactable blockchain protocol that achieves instant redaction confirmation by decoupling redaction voting from the consensus protocol. Their core idea is to have an external voting committee, selected independently from the consensus mechanism, approve or reject the redaction proposal. The voting committee is instantiated using verifiable random functions (VRFs) in Proof-of-Stake (PoS) environments or computational puzzles in Proof-of-Work (PoW) settings. The voters generate signed votes for the proposal and broadcast them to their peers. Votes are not immediately included on-chain; instead, a leader node aggregates votes that meet the quorum threshold. The leader constructs a redaction certificate including a Merkle tree of the votes, the list of signed approvals, and hash commitment to the original and redacted blocks. The certificate is embedded in the metadata of the next block, marking the redaction as final and allowing all nodes to verify it on-chain.

Thyagarajan et al. have proposed Reparo [178], a distinct redaction and repair mechanism that operates as a modular overlay on top of existing blockchain protocols. Reparo offers a distinct advantage by performing repairs on pre-existent data with no need to modify the underlying block structure or consensus algorithm. In concrete, any user can submit a repair proposal for a block which is subsequently deliberated on by a dynamically selected group of validators (also known as deciders). Repair proposals are in the form $rp := \langle (pt, x'), sp \rangle$, with pt being the block pointer, x' the new data, and sp the state update specifications (from st to st').

Miners are responsible for collecting repair requests and voting on them through a decision protocol (called deliberation). A miner mines the repair request (like other transactions) if it finds it worth discussion, then the deliberations ensue off-chain. They evaluate whether the proposal complies with the repair policy that dictates the constraints and requirements for repair operations and then they vote according to defined voting periods in the header of the new block.

Meng et al. [137] extend Deuber's voting model by introducing a conflict-resilient redactable blockchain architecture using Byzantine agreement consensus. This work aims to correct the critical limitation of Deuber's model, namely, the extended redaction latency and the lack of mechanisms to solve conflicting redactions of the same block. The work of [137] suggests selecting a conflict-free subset of requests per round to remove conflicting redactions. Then it uses committee-based consensus (inspired by Algorand [77]) to reach agreement on a set of redactions, providing instant finality after the BA round terminates. This method also decouples data redaction from new block proposals. The system continues to append blocks while redaction rounds can be held in parallel. Crucially, [137] handles up to one third of Byzantine miners in the network, even when the adversaries adaptively corrupt users or flood conflicting redactions.

AeRChain [127] introduces a redactable blockchain that operates fully anonymously in a permissionless PoW setting. Voter selection is done via a custom Proof-of-Work (cPoW), where miners generate redaction voting puzzles. Solving the puzzle grants the right to vote. The difficulty of the puzzle determines the voting weight. Voters sign their ballots using a *bLSAG* ring signature used in Monero [3], proving that they are legitimate while preserving voter anonymity. Nodes collect votes over a predefined voting period and

then the votes are aggregated and counted. If a quorum is met, a redacted block or transaction is finalized by replacing B_i with B_i*, and updating the list of old history storing prior Merkle roots for integrity. The network switches to the redacted view of the block. In particular, AeRChain supports anonymity while granting public verifiability of redactions with low latency.

Table 5.5: Comparison of voting-based redaction solutions.

Solution	Mechanisms	Gran.	Adversarial Model	Latency	Main Features
Deuber et al. [53]	Dual-hash voting via block references	Block	≤ 1/2 adversarial miners	Slow	- Poor redaction efficiency - Limited to non-monetary redactions - No conflict resilience
Marsalek et al. [131]	Dual-chain architecture (Standard + Correction)	Block	≤ 1/2 adversarial miners	Slow	- Partial monetary redactions supported - Heavy storage overhead - Compatible with legacy PoW chains
Xu et al. [192]	Stake-weighted VRF-based voting	Block	≤ 1/3 corrupted stake; static adversary	Fast	- Fast finality - Supports multiple redactions per block - Incentivized redactions
Reparo [178]	Off-chain deliberation + witness inclusion	Block, Tx, state	Same as underlying consensus	Medium	- Repairability of existing contents - Policy-based governance - Works with unmodified chains - Supports state corrections
Li et al. [124]	External VRF/PoW committee + redaction certificate	Block	≤ 1/2 computing power or stake	Fast	- Voting decoupled from consensus - Cryptographic redaction proofs - No built-in incentive layer
AeRChain [127]	Anonymous PoW-weighted voting (bLSAG + cPoW)	Block, Tx	≤ 1/2 computing power	Moderate	- Strong voter anonymity - Sybil resistance via PoW - Public verifiability via ring signatures
Meng et al. [137]	BFT committee agreement	Block	≤ 1/3 adaptive adversary	Fast	- Conflict-free redaction scheduling - Redaction decoupled from block creation - Secure against adaptive adversaries

PRBFPT [50] introduces a voting-based redactable blockchain design combined with chameleon hash assistance, in which all nodes share a public trapdoor, and editing rights are granted through on-chain smart contract governance. Rather than obscuring or distributing the trapdoor, PRBFPT publishes it directly in the block header, enabling all nodes to compute chameleon hash collisions independently. Security is preserved not through secrecy but through auditable voting and structural integrity constraints enforced at the protocol level.

All nodes share the same chameleon hash keys, including the trapdoor. Since redactions require consensus-backed authorization, PRBFPT treats the public trapdoor as non-threatening: it enables anyone to compute collisions, but only authorized edits are accepted during validation.

The redaction process in PRBFPT involves a smart contract–based locked voting scheme which determines whether a transaction should be edited or deleted. If the vote passes, all nodes execute a block insertion routine, creating a new block that replaces the offending data while preserving chain continuity. Importantly, chameleon hash collisions are used to maintain Merkle root and header hash consistency, so chain connectivity is preserved despite edits.

The dynamic membership of permissionless blockchains complicates the monitoring and synchronization of redactions. Chainknot [58] addresses this challenge through forced synchronization. Redaction proposals are submitted as transactions and evaluated by a dynamically formed voting committee derived from the underlying consensus mechanism. Upon approval, a designated redactor generates a new block containing the redacted content; this block becomes the new chain tip, and includes incentives for voters and the redactor. Crucially, nodes that refuse to adopt the redacted block lose the ability to verify or extend the chain. This approach enforces global convergence through incentivized consensus, unlike prior voluntary adoption schemes.

Table 5.5 provides a structured comparison of representative voting-based redaction protocols, highlighting their core mechanisms, redaction granularity, adversarial assumptions, latency characteristics, and distinctive design features.

Overall, voting-based redaction systems are characterized by several key properties. Redactions are user-initiated and then approved via consensus, involving a set of designated voters. These systems are generally coupled to the underlying consensus protocol, leveraging its security guarantees while introducing controlled mutability. Confirmed redactions are accompanied by cryptographic evidence, such as aggregated signatures, verifiable certificates, or witness proofs, supporting public verifiability and auditability of changes. While the majority operate at block-level granularity, only a few, notably Reparo [178] and AeRChain [127], support transaction- or state-level precision. Backward compatibility remains limited, only Reparo [178] stands apart as a backward-compatible layer for seamless integration with existing blockchain systems.

Summary. Consensus-based redaction schemes redefine immutability as a collective social contract rather than a purely cryptographic constraint. These methods embed redaction logic into the consensus layer, requiring a majority of network participants to approve and validate modification proposals through on-chain or off-chain voting. This category shines in decentralized environments, offering transparent, accountable, and scalable redaction mechanisms well-suited for public blockchains. By integrating redaction into the consensus mechanism, these schemes avoid trapdoor reliance and allow open participation, often with support for proposal justification and public auditing. However, consensus-based redaction faces key challenges. Namely, latency due to multi-phase voting rounds, security risks from vote manipulation or Sybil attacks, and the difficulty in maintaining semantic consistency after redactions (especially for smart

contracts or interlinked transactions). These systems are best suited for governance-driven environments where redaction decisions must be socially negotiated, and they represent an evolution towards self-regulating mutable blockchains.

5.4 Evaluation

We adopt the Bitcoin framework to implement the core functionalities of the evaluated redactable blockchains, with Bitcoin serving as our baseline. This approach allows us to establish a benchmark that objectively compares the performance of the redactable blockchain systems with the well-established Bitcoin framework. To this end, we implement three pioneering redaction solutions representing each respective classes (cryptographic [8], meta-transactions [155] and voting-based [53] redaction). We developed a custom discrete-event simulation framework. To the best of our knowledge, this represents the first open-source benchmarking suite specifically tailored for redactable blockchain systems. The framework is designed with modularity in mind, allowing the community to extend it with new redaction algorithms (e.g., zk-SNARK based schemes,VDF methods) by implementing standard interfaces for transaction processing and block validation.

Hardware Environment. All experiments were conducted on a physical workstation running Ubuntu 20.04 LTS. The machine was equipped with an Intel Xeon W-2123 Quad-Core processor (3.6 GHz base frequency) and 32 GB of DDR4 memory. All simulations were executed in a single-node environment to ensure deterministic control over network behavior and timing. The simulator extends the architecture of BlockSim, implemented in Python 3.10. It abstracts the network layer to eliminate transient internet fluctuations while preserving the logic of block propagation and verification. We prioritized a modular architecture to facilitate extensibility: cryptographic primitives are encapsulated in separate modules (e.g., a distinct Chameleon Hash interface), allowing researchers to seamlessly switch between different hashing methods or trapdoor implementations without refactoring the core consensus logic.

The experimental framework builds upon and extends our system for simulating redactable blockchains proposed in [63]. For the current work, we have significantly enhanced the framework to capture a broader range of system metrics, including throughput, latency, and energy consumption, while improving the scalability and configurability of the simulation environment. Our experimental framework yields valuable insight that may be extended to cover a wide spectrum of redactable blockchain systems. The source code, detailed configuration files, and raw data logs are publicly available at the GitHub repository: https://github.com/ImaneElabid/Redactable-blockchains

We evaluate the performance of blockchain systems in a controlled environment, initially assuming all nodes are honest. This approach enables us to isolate the core performance characteristics of each mechanism without the confounding effects of adversarial behavior. Unless stated otherwise, we adopt the following setup. Over a simulated period of 500,000 seconds, we construct a blockchain within a network of 1000 nodes. The results are averaged over 10 simulation runs for more precision. The simulation time is chosen to provide sufficient data to observe the performance distinctions between competing redaction methods. To closely mimic real-world conditions, we incorporate parameters such as block interval and propagation delays derived from the Bitcoin network model as shown is Table 5.7.

Our experiments utilize a chameleon hash construction based on prime factorization. In the centralized setting, a single node performs redaction, while the decentralized setting employs Shamir's secret sharing scheme [167] to distribute the trapdoor key among all miners that constitute 30% of network nodes. Voting-

based redaction also involves 30% of nodes as validators, with a threshold of $\frac{1}{2}$ of their votes required for modification within a period over 150 blocks. For the μchain, transaction sets contain an average of three versions. Note that a mutable transaction needs at least two versions: *active* and *nope* version. We chose an average of 3 to allow for potential mutations of some transactions within the set.

5.4.1 Experimental Results

In the first set of experiments, we focus on the core redaction process. Redaction time measurement excludes network latency or user request queuing, starting when validators actively work on the modification. We also track blockchain creation time to quantify the overhead associated with the structural changes required by different redaction methods.

Figure 5.6 depicts the performance time of different redaction methods. Note that in Figure 5.6 and Figure 5.8 the Y-axis has a logarithmic scale. From a blockchain creation and block generation perspective, the results reveal remarkably consistent performance across all redaction techniques in comparison with the baseline. The results indicate that the choice of redaction mechanism has minimal impact on the fundamental block creation process. The most notable observation is that chameleon hash with MPC shows slightly higher creation time, likely due to the additional computational overhead of multi-party computation protocols during the overall blockchain construction process. We also observe slight increases in voting-based and μchain methods, due to incorporating block state and multi-versioning respectively. This suggests that the complexity introduced by redaction operation has a minimal impact on regular block creation operations (without redactions).

More critically, the differences are more pronounced for the redaction time. The permissioned Chameleon Hash method demonstrates near-instantaneous redaction, facilitated by a single authorized entity. In its permissionless (MPC) variant, where trapdoors are secret-shared among multiple parties, the redaction time increases significantly (Further insights and analysis regarding this aspect will be discussed subsequently in Figure 5.7 and 5.12). Voting-based redaction remains approximatively 100× slower than centralized chameleon hash. This is a direct result of the extended voting periods, which are measured in terms of the mined blocks following the redaction initiation (in this experiment, the voting period Δt is configured to 10 blocks and the vote threshold ρ is 6 votes). We provide further analysis in Figure 5.8. The μchain's redaction time is moderately higher than the Chameleon Hash method, reflecting the additional steps of policy verification and transaction replacement within its structure.

Table 5.7: Simulation parameters and network configuration.

Parameters	Value
Simulation time	500000 s
Network size	1000 nodes
Miners fraction	30%
Simulation runs	10
Block interval	600 s
Block propagation delay	0.42 s

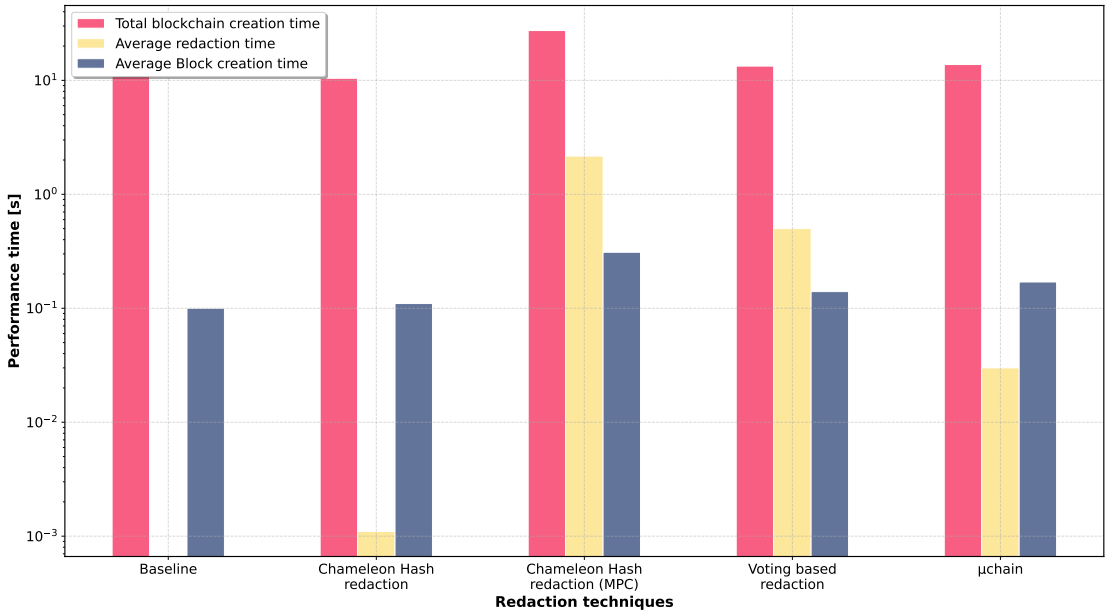


Figure 5.6: Blockchain creation and redaction time comparison for a network of 1,000 nodes (3% miners) processing 25k transactions under different redaction approaches. The permissioned chameleon hash-based approach achieves efficient redaction compared to voting-based and mutation-based approaches while incurring negligible overhead in block creation.

Exploring the extremely low performance of chameleon hash with MPC observed in the first experiment, figure 5.7 reveals that the redaction process exhibits exponential time complexity with the number of validators, demonstrating two distinct performance phases. In the initial phase (approximately 0 to 30% validator fraction) the redaction time remains relatively stable at around 0.5 seconds, suggesting that the secret sharing reconstruction overhead is manageable when only a subset of validators participate. The flat region indicates that the system can efficiently handle moderate validator participation without significant performance degradation. However, beyond the 30% threshold, the complexity transitions into an exponential growth phase, where redaction time increases dramatically as validator participation approaches 100% of the network size. Consequently, distributed chameleon hash schemes impose substantial performance penalties that become prohibitive as network size increases.

The second set of experiments evaluates the impact of voting periods and thresholds on redaction efficiency. We systematically vary both parameters to capture how they influence the time required for a redaction to be completed. Additionally, we measure the computation steps involved in redacting a block, such as block verification and vote addition. The results depicted in Figure 5.8 illustrates how the chosen voting period significantly influences redaction time within the voting-based scheme. Figure 5.8 showcases the contrast between the best and worst case scenarios for the redaction operation. The best-case scenario involves collecting votes from the ρ immediate blocks following the candidate block. While the worst-case scenario requires waiting the full voting period to gather enough votes. Moreover, the actual redaction time includes the verification time and the voting time, which determines the computational complexity of the redaction. Notably, the gap between actual redaction time and voting time emphasizes the impact of the distributed nature of blockchains. While the voting process takes place, mining continues in parallel, potentially delaying the collection of enough votes for a successful redaction. We conclude that longer voting periods likely increase the robustness of the redaction process, making it more difficult to manipulate

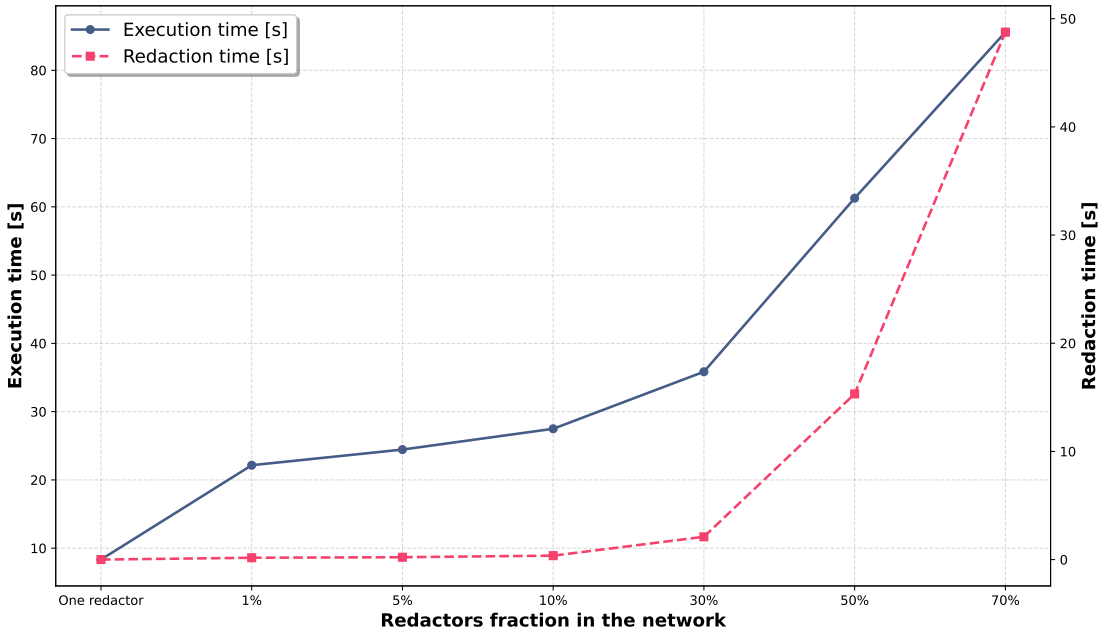


Figure 5.7: Impact of the number of validators on redaction time in a chameleon hash-based redaction system with secret key sharing, on a network of 1,000 nodes. Redaction time remains stable with up to 30% validator participation.

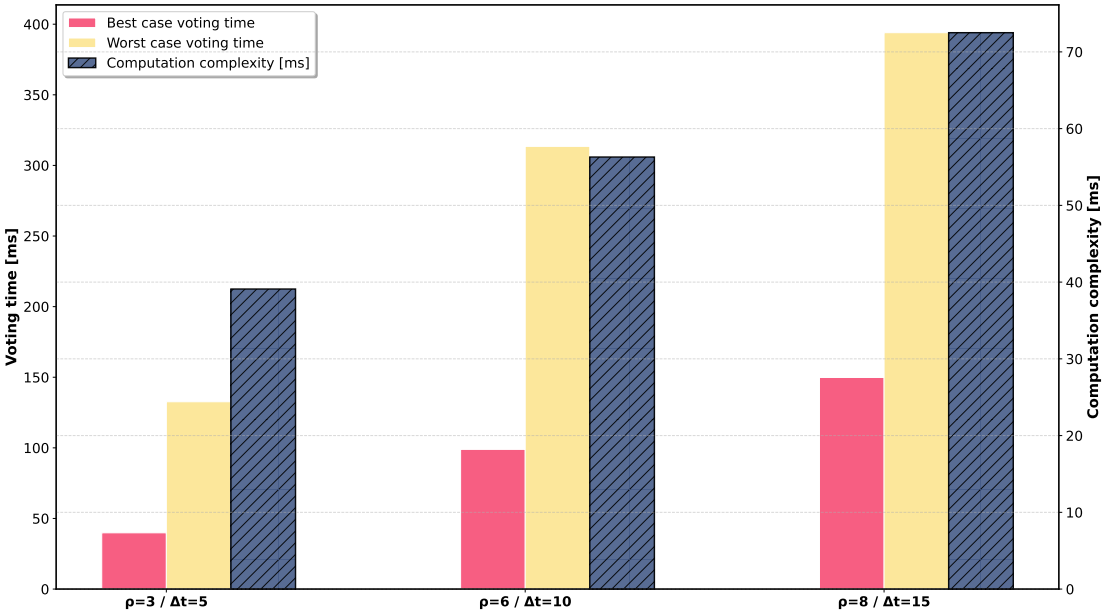


Figure 5.8: Best and worst-case voting period complexity versus actual redaction time in voting-based redaction for varying voting periods Δt and vote thresholds ρ, evaluated on a network of 1,000 nodes (3% validators) with 25k transactions. Higher ρ improves robustness but significantly increases redaction latency due to the distributed nature of voting.

votes. However, they come at the cost of longer redaction delays.

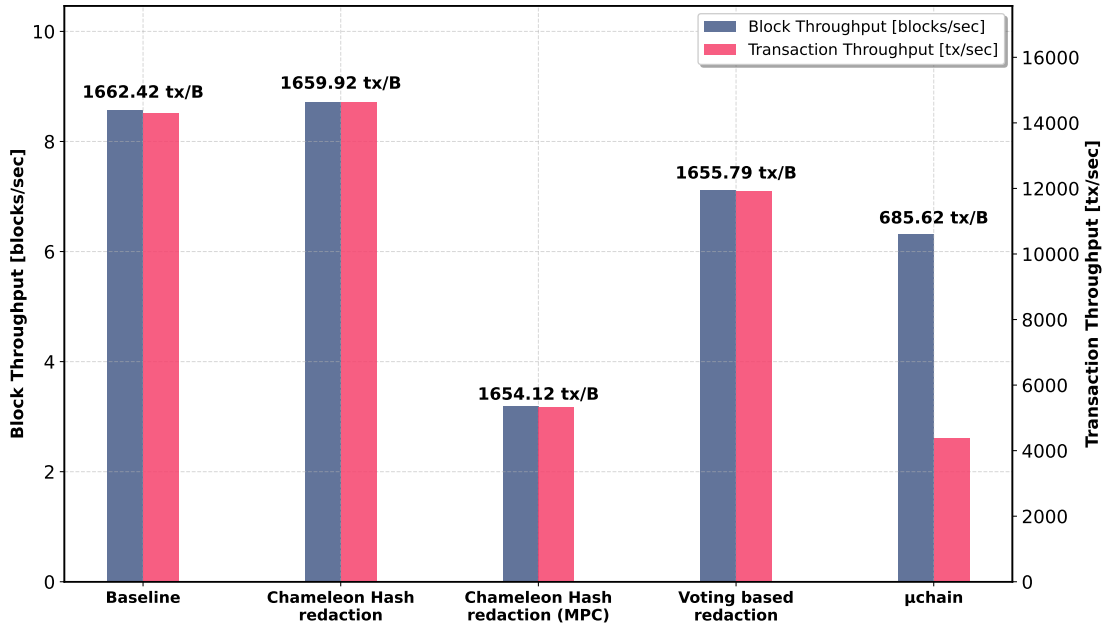


Figure 5.9: Transaction and block throughput comparison across redaction approaches processing 25k transactions. Permissioned chameleon hash achieves the highest throughput. Voting-based redaction shows a slight throughput reduction, while μchain exhibits a significant drop in transaction throughput.

In the following set of experiments, we measure the difference in block size among the evaluated redactable blockchain systems. Specifically, we measure the average number of transactions per block, block throughput, transaction throughput, and overall storage efficiency.

Our results in Figure 5.9 indicate a slight reduction in block throughput compared to the baseline for chameleon hash and voting approaches, while they maintain a similar number of transactions per block. Chameleon hash with MPC exhibits a dramatic throughput reduction, indicating that the multi-party computation overhead primarily affects block production rate. The MPC, though, keeps the same transaction density per block.

Most notably, μchain shows a substantial decrease in transactions per block with approximately 60% fewer transactions than other methods, while achieving comparable block throughput. To validate this observation, Figure 5.10 confirms that the chameleon hash and voting solutions maintain the same number of transactions per block as the baseline, whereas the μchain with different versioning configurations (μchain with 1~2, 1~3, and 1~4 tx/set configurations) has a substantially lower number of transactions per block.

The throughput penalty of the mutation-based solution is further compounded with a severe storage inefficiency, as demonstrated by Figure 5.11. The figure plots the cumulative number of transactions against the total storage used, revealing that a large portion of the allocated storage is wasted on maintaining transaction versions rather than storing new, unique transactions. The μchain [1–2 tx/set] configuration achieves only 41.1% storage efficiency, meaning nearly 60% of its allocated storage is occupied by versioning overhead rather than new transactions. The two other configurations perform even worse, incurring a substantial and inherent storage penalty as the versions increase.

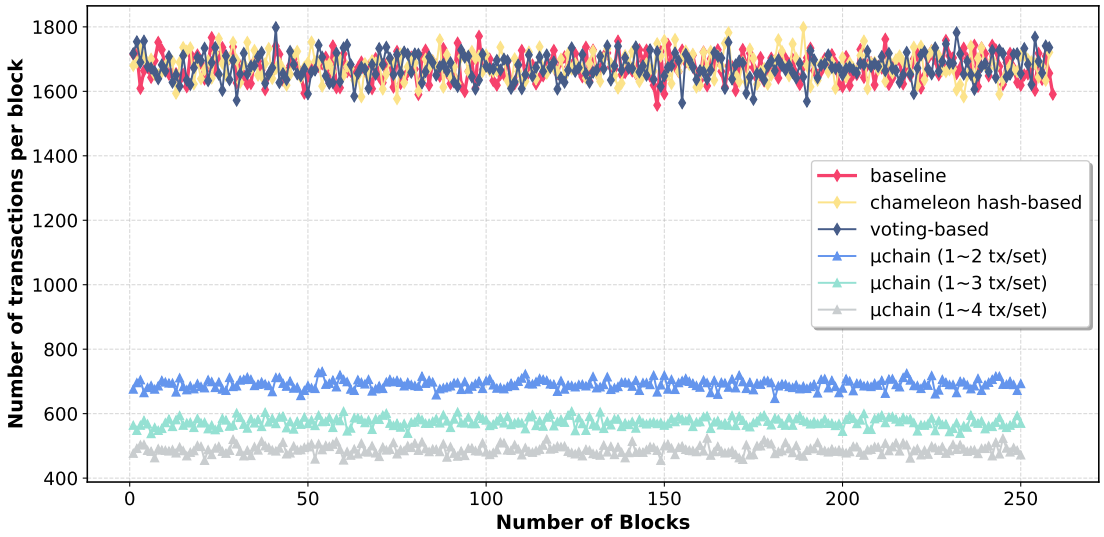


Figure 5.10: Transaction counts per block in different redaction techniques. The μchain configurations (e.g., 1~2 tx/set) indicate each set contains 1 to 2 transaction versions. A fixed "*nope*" transaction is included with every mutable transaction set, contributing to the overhead.

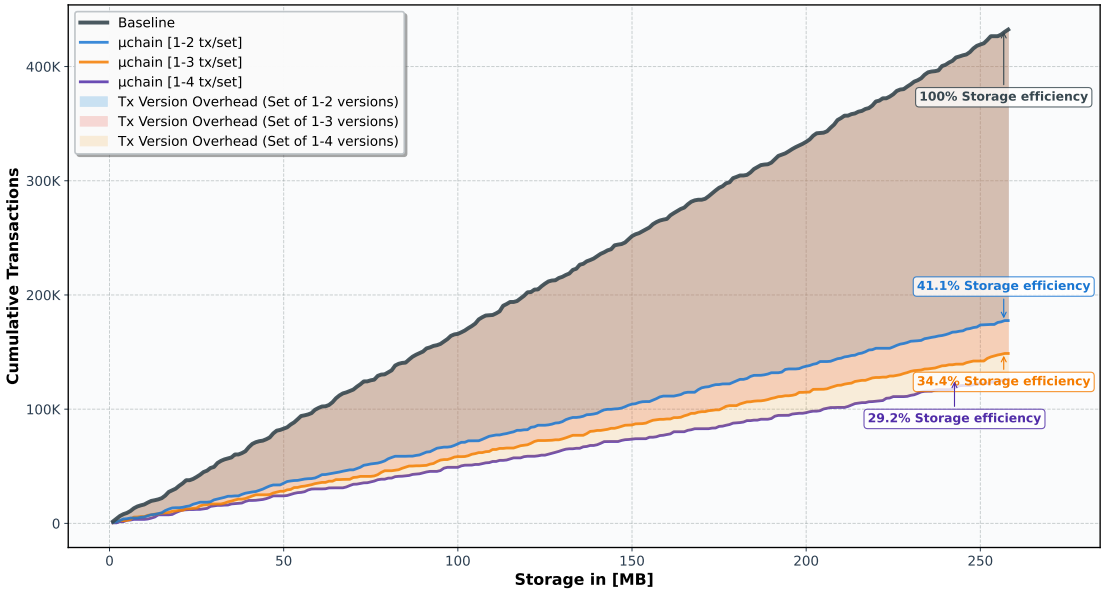


Figure 5.11: Cumulative transaction counts versus storage growth between the baseline and μchain configurations. The colored regions between the baseline (100% efficient) and μchain configurations (41.1%, 34.4%, and 29.2% efficient for 1~2, 1~3, and 1~4 version sets respectively) represent the wasted storage capacity dedicated to transaction versioning.

To examine the size overhead induced by the modifications to the block structure to build redactable blockchain systems. Table 5.9 showcases the new block fields sizes required in each of the three redaction approaches and their respective impacts on the total size overhead. Table 5.9 indicates the additional storage space required to implement these modifications compared to the baseline Bitcoin data structure.

The size overhead is defined as the additional load in the header and data field, as well as the overall impact on the entire block. Chameleon hash-based approach increases the size of the header by 40% due to the addition of hash randomness, but this results in only 0.0112% increase in the size of the entire block. In the case of the Voting-based approach, an extra field refers to the initial block state field y is included in the block header, resulting in a size overhead increase of approximately 40% of the header size. Additionally, votes are added to the data field, which leads to a negligible increase of 0.06% of the data size, resulting in a total size overhead of 0.0711%. In contrast, μchain has the largest size overhead of a 71% increase in the size of the data field due to the addition of transaction versions.

Our findings regarding block size variations are further supported by a granular analysis of the storage complexity across the evaluated systems. Table 5.9 quantifies the new block fields sizes required by each of the three redaction approaches and their resulting storage overhead. Table 5.9 indicates the additional storage required to implement these modifications compared to the Bitcoin model. The size overhead is defined as the additional load in the header and data field, as well as the overall impact on the entire block.

Table 5.9: Size overhead in the comparison to the baseline.

Component	Ateniese et al. [9]	Deuber et al. [53]	Puddu et al. [155]
Header	Hash randomness r (+40%)	Initial block state y (+40%)	-
Data	-	Votes (+0.06%)	Transaction set T (+71%)
Total	(+0.0112%)	(+0.0711%)	(+71%)

In this set of experiments, we evaluate the energy consumption of the redaction approaches relative to the baseline. Figure 5.9 reports the average energy consumption (in joules) for each approach when performing five redactions. Permissioned chameleon hash–based redaction exhibits the lowest overhead, consuming less than twice the baseline energy within a short redaction time. In contrast, the permissionless variant (MPC) requires more than 8× the baseline energy. For the voting redaction, although the redaction time is eight times longer than that of permissioned chameleon hash, the energy consumption is less than 1.2× higher, since most of the period is spent on lightweight voting rather than heavy computation. By comparison, μChain consumes about 1.6× more energy than permissioned chameleon hash, as redaction involves updating the block to select and set the appropriate transaction version as default.

5.4.2 Analytical Comparison and Trade-offs

In line with our experimental analysis, we present a comparative summary of key features, advantages, and drawbacks of the evaluated redaction techniques (i.e., chameleon hash-based, voting-based, and mutation-based redaction). Table 5.11 outlines representative solutions across dimensions such as network model, granularity, and storage overhead among others. This synthesis informs the design of redactable blockchain systems that better balance flexibility, efficiency, and security.

The chameleon hash-based approach, represented by Ateniese et al. [9], achieves minimal redaction latency through centralized intervention. However, it introduces a single point of failure that contradict blockchain decentralization principles.

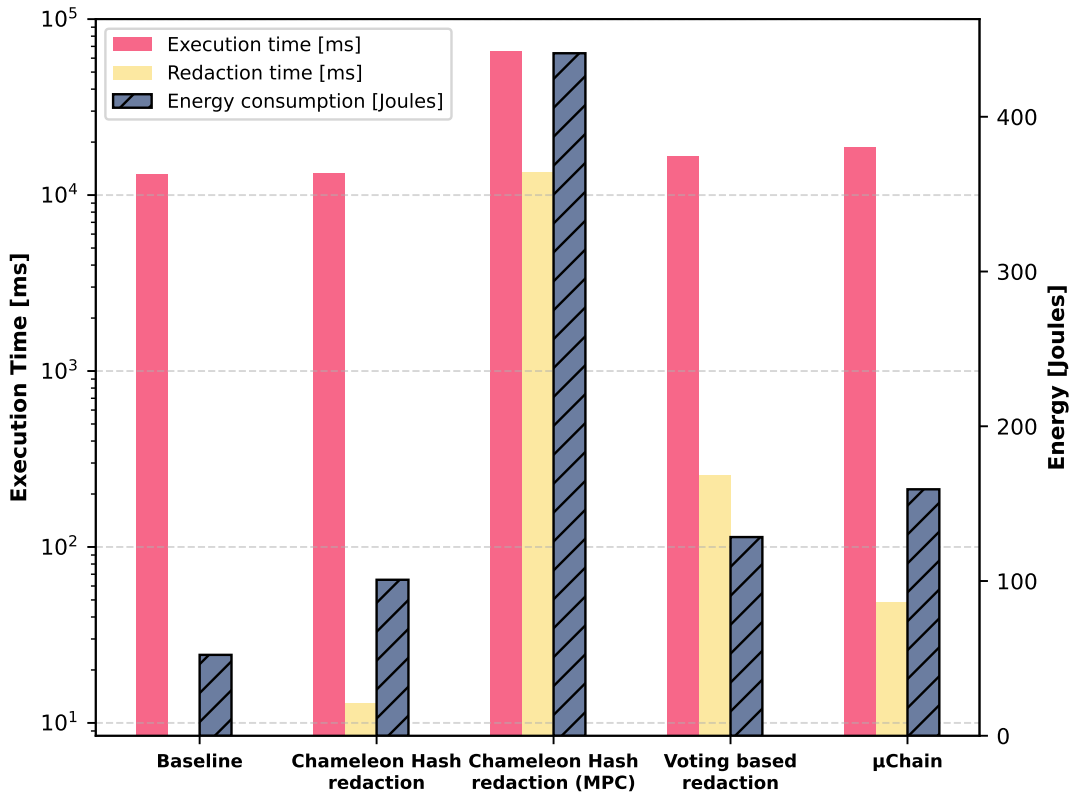


Figure 5.12: Average energy consumption and execution time of different redaction approaches relative to the baseline. Permissioned chameleon hash-based redaction is the most efficient, while permissionless chameleon hash with MPC incurs significantly higher energy consumption compared to other approaches.

More concretely, the chameleon hash approach [9], makes use of two major mechanisms to perform a reduction on block-level; chameleon hash and a secret sharing scheme [167] for key distribution. A secure MPC protocol is deployed in case of a consortium or a public network to secretly share the trapdoor key among the validators. This technique can be inefficient when the number of players involved in the MPC protocols is big, as it leads to reduced performance.

Additionally, chameleon hash-based methods offer low storage overhead but may face challenges related to key exposure [8], key management, and stateless redaction. Besides key management issues, a notable limitation of this approach is the lack of consideration for dependent transactions when performing redactions. This makes the system unsuitable for monetary transactions, as only stateless redactions are considered. All these elements makes this approach to be not deployable in existing blockchain systems.

Table 5.11: Comparison of redactable blockchain techniques.

Category	Cryptographic-based	Voting-based	Mutation-based
Core Mechanisms	CH, ZKP, RSA, Ring Signatures, MPC	Dual Hash Links, On-chain Voting, Logs	Multi-versions, Encrypted versions, Dependency tracking
Redaction Authority	Designated redactors	Network quorum via on-chain votes	Off-chain approval with enforced consistency
Governance Model	Centralized trusted setup or Distributed (e.g., DKG, TUCH)	Transparent, verifiable voting procedures	Hybrid trust: off-chain control, on-chain proof
Granularity	Block, transaction, or transaction-field level	Block-level only	Transaction-field level
Consistency Guarantee	Collision validity, State consistency	Voting confirmation and chain linking	Dependency-aware revalidation
Latency	Near-instantaneous (1 block)	High latency (multi-block)	Moderate, depends on propagation
Storage Overhead	Low to moderate: trapdoor, ZKP, proofs	Moderate: vote logs, dual headers	High: versioning, diffs, logs
Computation Overhead	Trapdoor ops, key and proof generation	Vote collection and quorum checks	Diff tracking, encryption/decryption
Auditability	Limited (trust-based assumptions)	High (votes are public)	Moderate (log-based)
Key Exposure Risk	High (persistent across schemes)	None (no keys)	Moderate (encryption keys)
Old Data Deletion	Obfuscated/rewritten	Marked via votes	Encrypted/rolled back
Best-fit Applications	IoT, constrained devices, industry blockchains	Public ledgers, community governance	GDPR systems, archival systems
Main Strengths	Fast edits, bounded trust, lightweight	Accountability, verifiability	Semantic edits, strong consistency
Key Limitations	Key setup/management, weak public traceability	High latency, scalability issues	Heavy storage, key management, complex updates

The redaction technique by Ateniese et al. [9] assumes that redactions occur only in exceptional cases. Editors are expected to act infrequently, not continuously. However, chameleon hash-based schemes are incompatible with existing blockchains as they require a different block header formats and replacing standard hash functions (e.g., SHA256) with chameleon hashes, limiting backward compatibility.

In contrast, the mutation technique introduces multiversioning, multi-key encryption, and secure multiparty computation to ensure transaction consistency. While effective for stateful redaction, these methods impose high storage overhead due to persistent version tracking and encryption data. Key management and efficiency remain major challenges. μChain [155], is primarily designed for private blockchains. Mutated

data is appended in subsequent blocks, and the old data is kept on-chain with no possibility to shrink the chain which result in a storage overhead.

To enable adoption in permissionless settings, secret sharing is used to distribute encryption keys among validators. However, this requires MPC protocols, introducing additional computational overhead. Moreover, μChain's modified data structures prevent backward compatibility with existing blockchains. Although μChain accounts for transaction dependencies to maintain currency balance after mutations, it suffers from policy limitations. Since redaction policies are defined by the sender, malicious users may insert illicit content without granting permission for removal. These solutions clearly fall short of the requirements of public blockchains.

Voting-based techniques, though slower due to extended consensus periods, offer public verifiability and enable content review during redaction. They raise important considerations, including voting duration, efficiency, incentive alignment, and support for monetary or non-monetary data. The scheme in [53] supports verifiability by retaining both original and redacted data links. It avoids complex cryptographic assumptions, enhancing suitability for public networks. The only added overhead is a voting layer during redaction. However, it also relies on modified block structure, limiting backward compatibility, and supports only stateless redactions. As a result, it cannot safely redact spendable content without risking consistency violations, making it unsuitable for payment systems.

5.5 Discussion and Open Problems

Our analysis reveals several critical research questions, including scalability, consistency, and adaptability among others. We believe that addressing these questions is critical for a large-scale adoption of redactable blockchain solutions in real-world applications.

5.5.1 Governance Illegibility and Privilege Abuse

A key governance challenge in redactable blockchains is the opacity of redaction privileges. Proposals vary from unilateral redactors with full cryptographic control to decentralized voting. For instance, Ateniese's foundational chameleon hash [9] assumes rare, trusted edits by a trapdoor holder but lacks visibility into frequency, intent, or misuse. Many similar systems rely on unverifiable edits and honest-majority validators, exposing them to collusion. Policy-based solutions [52, 180, 179, 98] improve transparency via access control and traitor tracing yet fail to establish penalty structures or reputation tracking for malicious validators.

Permissioned models [101, 201, 41] assign redaction authority to regulators or privileged nodes under fixed roles. While suitable for regulatory compliance (e.g., GDPR), this model introduces centralized trust dependencies. Open voting models [53, 192, 178] risk spam and abuse, while owner-only mutation models [155, 56] allow selective deletion of sensitive records. Systems like RedactChain [132] and μChain [155] introduce redaction windows, aligning with data minimization but limiting long-term erasure rights under GDPR.

In summary, while regulatory compliance motivates many redactable blockchain designs, most systems often prioritize cryptographic techniques over governance transparency. Future systems must integrate explicit privilege assignment, verifiable accountability, and robust enforcement within frameworks that meet legal standards.

5.5.2 Cost vs. Benefit Balance

Redaction operations introduce tangible overheads, including validation, voting, key reconstruction, and proof generation. These clash with profit-driven tasks like block production and transaction inclusion. Without explicit incentives, rational validators may ignore or delay redactions to preserve their mining advantage. Hence, a pay-for-performance strategy is essential to incentivize validator cooperation and ensure redacted data integrity.

While some proposals incorporate basic incentive models, most lack formal reward-aligned mechanisms. For example, Shen et al. [168] reward redaction approvals and penalize misbehavior. Similarly, CDEdit [41] requires users to buy tokens for redactions, compensating redactors accordingly. In voting-based schemes like Reparo [178] or [53], redaction fees exist but typically reward block proposers, not voters, leaving redactors uncompensated. This discourages validator engagement. Furthermore, systems with low-cost redaction requests are vulnerable to DoS and Sybil attacks, as adversaries may flood the network with malicious proposals. Xu et al. [192] introduces a incentive mechanism where validator rewards are weighted by stake, but still lacks penalties for non-participation.

In summary, incentive design for redactable blockchains remains a critical gap. Without mechanisms to compensate validators fairly and deter abuse, redaction systems risk limited adoption in permissionless environments where spontaneous participation cannot be assumed.

5.5.3 Conflicts Resolution and Inconsistencies

Another flaw in the design of redactable blockchains is the potential inconsistencies in the ledger that occur after redactions are made. The inability to handle conflicting edit requests with different objectives and granularities; When multiple users attempt to modify the same data, existing designs either process requests sequentially or reject them altogether. For instance, in intellectual property management, Alice may request removal of her property record, while Bob simultaneously asserts co-authorship. Without built-in conflict resolution, the system risks recording ambiguous or contradictory outcomes, undermining reliability and trust.

Additionally, redactable blockchains often lack version control to manage consistency across nodes. In the absence of canonical redacted states, nodes may hold divergent block histories. ”Zombie data” can persist on outdated chains, and nodes broadcasting stale versions degrade global consistency. Malicious actors may exploit this to reintroduce obsolete or sensitive data into the network. Some recent designs explicitly target post-redaction synchronization.Shen et al. [168] employ cryptographic accumulators to bind the active set of valid blocks, enabling nodes to detect superseded versions, while Wang et al. [186, 183] propose VDF-based redactable blockchains that enforces strong synchronization of redaction, ensuring all nodes maintain identical chain views post-modification. This problem is exacerbated in voting-based redaction systems. As shown by Dousti et al. [57], an adversary can redact a block that contains a vote for a pending redaction, erasing part of the vote history. This violates verifiability for new nodes, which can no longer validate the original redaction due to missing votes.

In summary, redactions are often treated as local edits, with the implicit assumption that all nodes will adopt the redacted chain. However, redactions constitute state transitions requiring global consistency guarantees. Each modification triggers cascading state recomputations, induces multiple version branches, and requires cross-node consistency proofs for global convergence. Without explicit mechanisms for conflict resolution, authenticated version propagation, and tamper-resistant vote records, redactable blockchains

remain vulnerable to divergence, rollback attacks, and unverifiable histories.

5.5.4 Scalability, Latency, and the Storage Minimization

Redactable blockchains aim to remove sensitive or obsolete data, yet most existing solutions paradoxically introduce considerable overhead in computation, latency, and storage. Chameleon-hash-based schemes incur costs from trapdoor key management, DKG, MPC, and ABE enforcement. ZKP-based protocols [23, 120] require intensive proof generation and verification. Lattice-based systems [191] add further load due to high-dimensional algebraic operations. Voting-based protocols [53, 178, 131] suffer from delayed finality caused by multi-round broadcasts and long voting periods, making them unsuitable for time-sensitive redactions under regulations like GDPR.

Many systems inflate ledger size through versioning, proofs, dependency tracking, and metadata. ReTRACe [147] and Wolverine [122] maintain redundant metadata, tamper-evident proofs, and revocation logs that persist after redactions. μChain [155] requires 71% heavier blocks to store multi-version transactions and encryption keys. ZKP-based systems [23, 120] add 40-60KB proofs per redaction. Voting schemes [53] introduce 40% header overhead for dual hashes and vote logs. On the other hand, MOF-BC [56] achieves 25% storage savings by summarizing outdated records. Low-latency protocols like Xu et al. [193] approve redactions within a single block, significantly reducing latency.

In summary, the substantial overheads in cryptographic processing, multi-round voting, and persistent metadata storage highlight the impracticality of uniform mutability guarantees across heterogeneous applications. Context-aware redaction schemes must adapt guarantees and optimizations to domain-specific requirements. Healthcare systems may tolerate overhead for regulatory auditability. IoT networks may enforce lightweight, time-bounded edits for efficiency. Central bank infrastructures might adopt post-quantum redactions where computational cost is secondary. Redactable blockchain design must align mutability semantics with application constraints while preserving distributed ledger trust properties.

5.6 Conclusion

Redactable blockchain technology is a complex, multidimensional field in its nascent stage. However, it is indispensable for blockchain to be viable in environments where privacy is a concern and resources are limited and regulatory compliance is required. In resource-constrained environments for example, where devices generate sensitive, high-volume data streams (e.g., energy consumption patterns, sensor telemetry), redaction offers dual advantages. First, by enabling the selective removal of sensitive identifiers, redaction implements privacy by data minimization to mitigate privacy risks inherent in permanent storage. Simultaneously, pruning obsolete or redundant data directly mitigates the storage and bandwidth burdens that would otherwise render blockchain participation prohibitive for resource-constrained devices.

Building on this premise, this chapter introduced a comprehensive taxonomy of various redaction classes, including cryptographic-based, mutation-based, and voting-based solutions. Additionally, we present a benchmarking framework that evaluates redactable blockchain solutions across a range of dimensions, including security, performance, governance, and scalability. We have demonstrated that no single technique dominates across all metrics. Instead, our comparative evaluation reveals that the suitability of a redaction method is fundamentally contingent upon application-specific requirements. Factors like the trust model, how often data needs to be updated, and regulatory rules are critical in selecting the most suitable technique. Ultimately, the chapter establishes a rigorous baseline for evaluating redactable protocols and sets the stage

for extensible experimentation, guiding both practitioners and researchers toward informed design choices in an area where standardized methodologies remain limited.

6 Conclusions and Future Work

6.1 Summary

This thesis has addressed the fundamental barriers preventing the widespread adoption of Blockchain systems, namely the prohibitive costs of consensus, the exclusion of resource-constrained devices, and the rigidity of immutability in a dynamic world. Throughout this work we confronted a critical juncture that blockchains must evolve from static ledgers into smart infrastructures that adapt to context, resource availability, and societal norms. More critically, we argued that the celebrated *Blockchain Trilemma* is not a rigid physical constraint about blockchain systems, but rather a design challenge that could progressively be mitigated through architectural considerations about communication, participation, and data governance. To this end, we have built a comprehensive framework that promotes the evolution of blockchain technology from static ledgers to adaptive, context-aware infrastructures that adapt to context, resource availability, and societal norms. Our work is structured around three complementary contributions:

Our first contribution, LighTx, tackles the scalability problem at its root: inefficient transaction dissemination. Through Byzantine Reliable Broadcast primitive and Proof-of-Bandwidth reputation system, we demonstrated that secure transaction propagation does not require global consensus or costly leader election. Instead, it can be achieved with logarithmic communication complexity, enabling high throughput and low latency in public, adversarial networks.

Our second contribution, PocketChain, directly addresses the trilemma's trade-offs in heterogeneous environments. The $\mathscr{P}ccp$ protocol introduces a novel, resource-aware consensus model where a device's role is dynamically aligned with its capabilities. This architecture fundamentally redefines accessibility, transforming billions of smartphones and IoT devices from passive clients into active, stake-holding participants in the consensus process.

Our third contribution, the systematic study on Redactable Blockchains, addresses the societal and regulatory friction caused by strict immutability. By providing a refined taxonomy and a benchmark of techniques, we demonstrated that controlled mutability when implemented with cryptographic transparency and auditibility can be a feature that strengthens the technology's applicability in privacy-sensitive and regulated industries like healthcare and finance.

This research shows that scalability, mobility, and privacy are not mutually exclusive but can be integrated at different layers of the stack (e.g., network, consensus, data) to collectively resolve the trilemma's constraints across multiple levels. At the device level, PocketChain turns smartphones into nodes, to

act as blockchain nodes, reducing dependency on energy-intensive specialized hardware and opening opportunities for sustainable deployments for new use cases. At the network level, LighTx challenges entrenched consensus assumptions by showing that secure dissemination can be achieved without energy waste, global quorums, or costly leader elections, while redaction techniques provide the formal and practical foundations for sophisticated on-chain data governance. At the societal level, the combined contributions point to blockchains that integrate seamlessly into real-life applications. For instance, efficiency at the communication layer ensures transaction systems can scale to public payment networks without prohibitive energy costs; inclusivity at the consensus layer allows mobile devices to secure supply-chain systems or IoT infrastructures without delegating trust to powerful actors; and mutability at the data layer supports compliance and accountability in domains such as healthcare, digital identity, or finance. By making the technology accessible to everyday devices, efficient for real-time transactions, and handling sensitive information, we present a paradigm for a more inclusive and trustworthy digital future.

In synthesis, this thesis demonstrably advances the state of the art by pushing blockchain design closer to systems that are lightweight enough to be carried in a pocket, robust enough for financial institutions, and flexible enough to meet diverse human and societal needs.

6.2 Limitations

While this thesis advances the state of the art in scalable and resource-adaptive blockchain systems, several limitations must be acknowledged to properly frame the scope of the contributions.

First, regarding LighTx. While demonstrating efficient transaction dissemination, operates under specific assumptions that present avenues for further investigation.

- Incentive Model Scrutiny: The current Proof-of-Bandwidth model effectively mitigates Sybil attacks but does not fully explore a granular incentive structure. Future work should rigorously model economic incentives to ensure long-term participation and security against more sophisticated adversaries.
- Network Topology Dependence: The logarithmic complexity and performance of the sBRB primitive are optimal in well-connected, peer-to-peer networks. Its efficiency in highly adversarial network conditions, such as pervasive partitioning or targeted eclipse attacks, requires further empirical analysis.
- Integration with State-Machine Replication (SMR): LighTx solves propagation but deliberately decouples it from ordering. A critical next step is a seamless and secure integration of this dissemination layer with a full consensus protocol for state machine replication to form a complete blockchain stack.

Second, concerning PocketChain. PocketChain's resource-aware consensus is a novel step towards inclusivity, yet its practical deployment introduces several challenges.

- Hardware Heterogeneity and Benchmarking: While $\mathscr{P}ccp$ is designed for heterogeneity, the performance model relies on abstract resource metrics. A more extensive real-world benchmarking campaign across a wider array of devices (from low-end IoT sensors to flagship smartphones) is needed to refine the role-assignment algorithm and validate the energy efficiency claims at scale.

- Network Churn and Availability: The dynamic nature of mobile devices, characterized by frequent disconnections and IP changes, poses a challenge to network stability. Enhancing the protocol to be highly resilient to churn without sacrificing liveness or security guarantees is a non-trivial and essential extension. PocketChain's resource-aware consensus is a novel step towards inclusivity, yet its practical deployment introduces several challenges.

Third, with respect to Redactable Blockchains. The taxonomy and benchmarking framework proposed in this thesis provide structure to an emerging field, but the evaluation remains bounded by methodological constraints. Specifically, the experimental study was necessarily limited to a representative subset of techniques, each representing a major category. Although this design captures high-level trade-offs, it does not fully reflect the nuanced implementation details or optimizations of other systems within each category. The findings should therefore be interpreted as indicative of a class's potential, not a definitive evaluation of every solution.

Finally, at the thesis level. The three contributions, while complementary, have been developed and evaluated as distinct research artifacts. Although we have argued for their conceptual synergy, a full-stack integration remains outside the scope of this work.

6.3 Future Directions

As the contributions of this thesis address significant hurdles to blockchain adoption, they also open a multitude of potential avenues for future research. The limitations outlined in the previous section serve as a direct point of departure for these avenues. The following perspectives are organized to progress from specific extensions of this work to broader, cross-disciplinary visions for the future of decentralized systems.

An immediate next step is the development of a unified data state management protocol. The storage model introduced in PocketChain slightly introduced how hierarchical summarization and archival policies can mitigate blockchain growth in resource-constrained environments. Similarly, redactable blockchains demonstrate that immutability can be reconciled with accountability through controlled mutability. The combination of the concept of storage optimization and the knowledge gained from our systematic study of redactable blockchains, provides the the conceptual and technical knowledge required to implement data and state management mechanisms. We envision a unified framework in which advanced redaction techniques are adapted to address storage challenges, integrated with compression mechanisms to create a lightweight yet verifiable chain of state. This direction represents a form of targeted forgetting that is essential for long-term sustainability.

On another level, while initial proofs-of-concept are promising, future work must involve large-scale simulations in networks, and ultimately, limited pilot deployments in live payment or IoT infrastructures. Extensive evaluation on networked testbeds under Byzantine conditions that mirror the real internet is critical to quantify performance, energy consumption, and resilience under adversarial, global-scale conditions. Such evaluation would also provide the necessary insights for refining system parameters, optimizing protocols for heterogeneous environments, and ensuring robustness when scaling from controlled experiments to societal infrastructures.

Moving forward, the most promising research direction emerging from this work is the vision of a Self-Adapting Blockchain that continuously reconfigure themselves in response to environmental and societal

contexts. Building on PocketChain's resource-aware consensus, future systems could integrate predictive models of device availability, network conditions, and workload distribution, enabling elastic consensus and adaptive role allocation. The formalization of guarantees for such adaptive protocols, including liveness, safety, and fairness, under reconfiguration, remains an open research challenge.

Finally, the ultimate vision is to move toward cross-disciplinary, application-aware blockchain infrastructures. Rather than treating blockchains as generic ledgers, future research should aim at modular architectures tailored to specific domains such as healthcare, digital identity, or supply-chain management.

Bibliography

[1] Shams Mhmood Abd Ali, Mohd Najwadi Yusoff, and Hasan Falah Hasan. "Redactable Blockchain: Comprehensive Review, Mechanisms, Challenges, Open Issues and Future Research Directions." In: *Future Internet* 15.1 (2023), p. 35.

[2] Shashank Agrawal and Melissa Chase. "FAME: Fast attribute-based message encryption." In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 665–682.

[3] Kurt M Alonso et al. "Zero to monero." In: ().

[4] Erfan Andesta, Fathiyeh Faghih, and Mahdi Fooladgar. "Testing smart contracts gets smarter." In: *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE. 2020, pp. 405–412.

[5] *Art. 17 GDPR - right to erasure ('right to be forgotten')*. Nov. 2018. URL: https://gdpr.eu/article-17-right-to-be-forgotten/.

[6] Kondapally Ashritha, M Sindhu, and KV Lakshmy. "Redactable blockchain using enhanced chameleon hash function." In: *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*.

[7] James Aspnes. "Notes on theory of distributed systems." In: *arXiv preprint arXiv:2001.04235* (2020).

[8] Giuseppe Ateniese and Breno de Medeiros. "On the key exposure problem in chameleon hashes." In: *International Conference on Security in Communication Networks*. Springer. 2004, pp. 165–179.

[9] Giuseppe Ateniese et al. "Redactable blockchain–or–rewriting history in bitcoin and friends." In: *2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE. 2017, pp. 111–126.

[10] Gennaro Avitabile et al. "Data Redaction in Smart-Contract-Enabled Permissioned Blockchains." In: *CEUR WORKSHOP PROCEEDINGS*. Vol. 3791. CEUR-WS. 2024.

[11] Leo Maxim Bach, Branko Mihaljevic, and Mario Zagar. "Comparative analysis of blockchain consensus algorithms." In: *2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO)*.

[12] Adam Back et al. "Hashcash-a denial of service counter-measure." In: (2002).

[13] Adam Back et al. "Enabling blockchain innovations with pegged sidechains." In: *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* 72 (2014), pp. 201–224.

[14] Marshall Ball et al. "Proofs of useful work." In: *Cryptology ePrint Archive* (2017).

[15] Joshua Baron et al. "Communication-optimal proactive secret sharing for dynamic groups." In: *Applied Cryptography and Network Security: 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*. Springer. 2016, pp. 23–41.

[16] Salman A Baset and Henning Schulzrinne. "An analysis of the skype peer-to-peer internet telephony protocol." In: *arXiv preprint cs/0412017* (2004).

[17] Robert Basmadjian et al. "On the advantages of P2P ML on mobile devices." In: *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*. 2022.

[18] Juan Benet. "Ipfs-content addressed, versioned, p2p file system." In: *arXiv preprint arXiv:1407.3561* (2014).

[19] John Bethencourt, Amit Sahai, and Brent Waters. "Ciphertext-policy attribute-based encryption." In: *2007 IEEE symposium on security and privacy (SP'07)*.

[20] *Bitcoin blockchain size 2009-2025 | Statista — statista.com*. https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/. [Accessed 15-03-2025]. February 2025.

[21] *Bitcoin energy comparison, by country 2025 | Statista — statista.com*. https://www.statista.com/statistics/881522/bitcoin-energy-consumption-relative-to-select-countries/. [Accessed 15-03-2025]. January 2025.

[22] *Blockchain.com | Charts - Blockchain Size (MB) — blockchain.com*. https://www.blockchain.com/explorer/charts/blocks-size. [Accessed 01-05-2025].

[23] Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. "Towards data redaction in bitcoin." In: *IEEE Transactions on Network and Service Management* 19.4 (2022), pp. 3872–3883.

[24] Gabriel Bracha. "Asynchronous Byzantine agreement protocols." In: *Information and Computation* 75.2 (1987), pp. 130–143.

[25] Gabriel Bracha and Sam Toueg. "Asynchronous consensus and broadcast protocols." In: *Journal of the ACM (JACM)* 32.4 (1985), pp. 824–840.

[26] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." In: *Computer networks and ISDN systems* 30.1-7 (1998), pp. 107–117.

[27] Benedikt Bünz et al. "Flyclient: Super-light clients for cryptocurrencies." In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2020, pp. 928–946.

[28] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform." In: *white paper* 3.37 (2014), pp. 2–1.

[29] Vitalik Buterin. *An incomplete guide to rollups*. [Accessed 24-05-2025]. 2021.

[30] Vitalik Buterin et al. “Ethereum white paper.” In: *GitHub repository* 1.22-23 (2013), pp. 5–7.

[31] Vitalik Buterin et al. *Sharding FAQ*. https://vitalik.eth.limo/general/2017/12/31/sharding_faq.html. [Accessed 02-05-2024].

[32] Vitalik Buterin. “The meaning of decentralization.” In: *Medium* (2017). URL: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274.

[33] Vitalik Buterin and Virgil Griffith. “Casper the friendly finality gadget.” In: *arXiv preprint arXiv:1710.09437* (2017).

[34] Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. *Introduction to reliable and secure distributed programming*. Springer Science & Business Media, 2011.

[35] Christian Cachin and Marko Vukolić. “Blockchain consensus protocols in the wild.” In: *arXiv preprint arXiv:1707.01873* (2017).

[36] Jan Camenisch et al. “Chameleon-hashes with ephemeral trapdoors.” In: *IACR International Workshop on Public Key Cryptography*. Springer. 2017.

[37] Fran Casino et al. “Immutability and decentralized storage: An analysis of emerging threats.” In: *IEEE Access* 8 (2019), pp. 4737–4744.

[38] Miguel Castro, Barbara Liskov, et al. “Practical byzantine fault tolerance.” In: *OsDI*. Vol. 99. 1999. 1999, pp. 173–186.

[39] Tushar Deepak Chandra and Sam Toueg. “Unreliable failure detectors for reliable distributed systems.” In: *Journal of the ACM (JACM)* 43.2 (1996), pp. 225–267.

[40] Dimitris Chatzopoulos et al. “Mneme: A mobile distributed ledger.” In: *Ieee Infocom 2020-Ieee Conference On Computer Communications*. IEEE. 2020, pp. 1897–1906.

[41] Xiaofeng Chen and Ying Gao. “Cdedit: A highly applicable redactable blockchain with controllable editing privilege and diversified editing types.” In: *arXiv preprint arXiv:2205.07054* (2022).

[42] Lichen Cheng et al. “Polynomial-based modifiable blockchain structure for removing fraud transactions.” In: *Future generation computer systems* 99 (2019), pp. 154–163.

[43] Bram Cohen. “Incentives build robustness in BitTorrent.” In: *Workshop on Economics of Peer-to-Peer systems*. Vol. 6. 2003, pp. 68–72.

[44] Bram Cohen and Krzysztof Pietrzak. “The chia network blockchain.” In: *White Paper, Chia. net* 9.2 (2019).

[45] *Competition and Data — privacyinternational.org*. https://privacyinternational.org/explainer/2293/competition-and-data. [Accessed 23-09-2025]. 26-09-2018.

[46] Lin William Cong, Zhiguo He, and Jiasun Li. “Decentralized mining in centralized pools.” In: *The Review of Financial Studies* 34.3 (2021), pp. 1191–1235.

[47] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[48] Flaviu Cristian. “Reaching agreement on processor-group membrship in synchronous distributed systems.” In: *Distributed Computing* 4.4 (1991), pp. 175–187.

[49] Kyle Croman et al. “On scaling decentralized blockchains.” In: *International conference on financial cryptography and data security*. Springer. 2016, pp. 106–125.

[50] Weiqi Dai et al. “PRBFPT: A practical redactable blockchain framework with a public trapdoor.” In: *IEEE Transactions on Information Forensics and Security* 19 (2024), pp. 2425–2437.

[51] Primavera De Filippi and Smari McCarthy. “Cloud computing: Centralization and data sovereignty.” In: *European Journal of Law and Technology* 3.2 (2012).

[52] David Derler et al. “Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based.” In: *Cryptology ePrint Archive* (2019).

[53] Dominic Deuber, Bernardo Magri, and Sri Aravinda Krishnan Thyagarajan. “Redactable blockchain in the permissionless setting.” In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 124–138.

[54] Whitfield Diffie and Martin E Hellman. “Multiuser cryptographic techniques.” In: *Proceedings of the June 7-10, 1976, national computer conference and exposition*. 1976, pp. 109–112.

[55] Dogecoin. *Do Only Good Everyday — dogecoin.com*. https://dogecoin.com/. [Accessed 12-04-2025].

[56] Ali Dorri, Salil S Kanhere, and Raja Jurdak. “MOF-BC: A memory optimized and flexible blockchain for large scale networks.” In: *Future Generation Computer Systems* 92 (2019), pp. 357–373.

[57] Mohammad Sadeq Dousti and Alptekin Küpçü. “Moderated redactable blockchains: A definitional framework with an efficient construct.” In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2020.

[58] Junke Duan et al. “Chainknot: Redactable blockchain protocol with forced synchronization.” In: *IEEE Transactions on Network Science and Engineering* 11.3 (2024), pp. 3091–3104.

[59] Sisi Duan, Michael K Reiter, and Haibin Zhang. “BEAT: Asynchronous BFT made practical.” In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 2028–2041.

[60] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. “Consensus in the presence of partial synchrony.” In: *Journal of the ACM (JACM)* 35.2 (1988), pp. 288–323.

[61] Cynthia Dwork and Moni Naor. “Pricing via processing or combatting junk mail.” In: *Annual international cryptology conference*. Springer. 1992, pp. 139–147.

[62] Stefan Dziembowski et al. “Proofs of space.” In: *Annual Cryptology Conference*. Springer. 2015, pp. 585–605.

[63] Imane El Abid and Yahya Benkaouz. "A comparative analysis of blockchain redaction techniques." In: *2024 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE. 2024, pp. 93–102.

[64] Imane El Abid and Yahya Benkaouz. "LighTx: a lightweight transactions transfer system." In: *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2021, pp. 1–3.

[65] Imane El Abid, Yahya Benkaouz, and Ahmed Khoumsi. "LighTx: A Lightweight Proof-of-Bandwidth Transactions Transfer System." In: *Networked Systems: 9th International Conference, NETYS, Proceedings*. Springer. 2021, pp. 144–160.

[66] Paul Erdős and Alfréd Rényi. "On the evolution of random graphs." In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960), pp. 17–60.

[67] Ittay Eyal et al. "Bitcoin-ng: A scalable blockchain protocol." In: *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*. 2016, pp. 45–59.

[68] Efat Fathalla et al. "Redactable Distributed Ledgers: A Survey." In: *Distributed Ledger Technologies: Research and Practice* (2023).

[69] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. "Impossibility of distributed consensus with one faulty process." In: *Journal of the ACM (JACM)* 32.2 (1985), pp. 374–382.

[70] Ethereum Foundation. *The Merge — ethereum.org*. https://ethereum.org/en/roadmap/merge/. [Accessed 16-03-2025].

[71] Keke Gai, Meikang Qiu, and Xiaotong Sun. "A survey on FinTech." In: *Journal of Network and Computer Applications* 103 (2018), pp. 262–273.

[72] David Galindo et al. "Fully distributed verifiable random functions and their application to decentralised random beacons." In: *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2021, pp. 88–102.

[73] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. "The bitcoin backbone protocol: Analysis and applications." In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2015.

[74] Adem Efe Gencer et al. "Decentralization in bitcoin and ethereum networks." In: *International conference on financial cryptography and data security*. Springer. 2018, pp. 439–457.

[75] *General Data Protection Regulation (GDPR) – Official Legal Text — gdpr-info.eu*. https://gdpr-info.eu/. [Accessed 22-Jun-2023].

[76] Arthur Gervais et al. "On the security and performance of proof of work blockchains." In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 3–16.

[77] Yossi Gilad et al. "Algorand: Scaling byzantine agreements for cryptocurrencies." In: *Proceedings of the 26th symposium on operating systems principles*. 2017, pp. 51–68.

[78] Seth Gilbert and Nancy Lynch. "Perspectives on the CAP Theorem." In: *Computer* 45.2 (2012), pp. 30–36.

[79] *Global Payments & Financial Solutions for Businesses — ripple.com*. https://ripple.com/. [Accessed 12-04-2025].

[80] Oded Goldreich. "Computational complexity: a conceptual perspective." In: *ACM Sigact News* 39.3 (2008), pp. 35–39.

[81] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[82] Dima Grigoriev and Vladimir Shpilrain. "RSA and redactable blockchains." In: *International Journal of Computer Mathematics: Computer Systems Theory* 6.1 (2021), pp. 1–6.

[83] Rachid Guerraoui et al. "AT2: asynchronous trustworthy transfers." In: *arXiv preprint arXiv:1812.10844* (2018).

[84] Rachid Guerraoui et al. "Consensus in asynchronous distributed systems: A concise guided tour." In: *Advances in Distributed Systems: Advanced Distributed Computing: From Algorithms to Systems*. Springer, 2002, pp. 33–47.

[85] Rachid Guerraoui et al. "Scalable byzantine reliable broadcast (extended version)." In: *arXiv preprint arXiv:1908.01738* (2019).

[86] Rachid Guerraoui et al. "The consensus number of a cryptocurrency." In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 2019, pp. 307–316.

[87] P Krishna Gummadi, Stefan Saroiu, and Steven D Gribble. "A measurement study of Napster and Gnutella as examples of peer-to-peer file sharing systems." In: *ACM SIGCOMM Computer Communication Review* 32.1 (2002), pp. 82–82.

[88] Vassos Hadzilacos and Sam Toueg. *A modular approach to fault-tolerant broadcasts and related problems*. Tech. rep. Cornell University, 1994.

[89] Jawad Haj-Yahya et al. "Techniques for reducing the connected-standby energy consumption of mobile devices." In: *2020 IEEE International Symposium on High Performance Computer Architecture*. IEEE. 2020, pp. 623–636.

[90] MyungJoo Ham, Inki Dae, and Chanwoo Choi. "{LPD}: Low Power Display Mechanism for Mobile and Wearable Devices." In: *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 2015, pp. 587–598.

[91] Jun Wook Heo et al. "Blockchain data storage optimisations: a comprehensive survey." In: *ACM Computing Surveys* 56.7 (2024), pp. 1–27.

[92] Maurice Herlihy. "Wait-free synchronization." In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 13.1 (1991), pp. 124–149.

[93] Maurice P Herlihy and Jeannette M Wing. "Linearizability: A correctness condition for concurrent objects." In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 12.3 (1990), pp. 463–492.

[94] Max Hort et al. "A survey of performance optimization for mobile applications." In: *IEEE Transactions on Software Engineering* 48.8 (2021), pp. 2879–2904.

[95] Huawei Huang et al. "A survey of state-of-the-art on blockchains: Theories, modelings, and tools." In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–42.

[96] Ke Huang et al. "Achieving intelligent trust-layer for Internet-of-Things via self-redactable blockchain." In: *IEEE Transactions on Industrial Informatics* 16 (2019).

[97] Ke Huang et al. "Building redactable consortium blockchain for industrial Internet-of-Things." In: *IEEE Transactions on Industrial Informatics* 15.6 (2019), pp. 3670–3679.

[98] Ke Huang et al. "Scalable and redactable blockchain with update and anonymity." In: *Information Sciences* 546 (2021), pp. 25–41.

[99] Damien Imbs and Michel Raynal. "Trading off t-resilience for efficiency in asynchronous byzantine reliable broadcast." In: *Parallel Processing Letters* 26.04 (2016), p. 1650017.

[100] Markus Jakobsson and Ari Juels. "Proofs of work and bread pudding protocols." In: *Secure Information Networks: Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99) September 20–21, 1999, Leuven, Belgium*. Springer. 1999, pp. 258–272.

[101] Yanxue Jia et al. "Redactable blockchain supporting supervision and self-management." In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 2021.

[102] Prasad Jogalekar and Murray Woodside. "Evaluating the scalability of distributed systems." In: *IEEE Transactions on parallel and distributed systems* 11.6 (2002), pp. 589–603.

[103] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. "The eigentrust algorithm for reputation management in p2p networks." In: *Proceedings of the 12th international conference on World Wide Web*. 2003, pp. 640–651.

[104] Ghassan Karame, David Gubler, and Srdjan Čapkun. "On the security of bottleneck bandwidth estimation techniques." In: *International Conference on Security and Privacy in Communication Systems*. Springer. 2009, pp. 121–141.

[105] Anne-Marie Kermarrec and Maarten Van Steen. "Gossiping in distributed systems." In: *ACM SIGOPS operating systems review* 41.5 (2007), pp. 2–7.

[106] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. "Non-interactive proofs of proof-of-work." In: *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers*. Springer. 2020, pp. 505–522.

[107] Aggelos Kiayias et al. "Ouroboros: A provably secure proof-of-stake blockchain protocol." In: *Annual international cryptology conference*. Springer. 2017, pp. 357–388.

[108] Kazumasa Koitani, Go Hasegawa, and Masayuki Murata. "End-to-end measurement of hop-by-hop available bandwidth." In: *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*. IEEE. 2014, pp. 17–24.

[109] Eleftherios Kokoris-Kogias et al. “Omniledger: A secure, scale-out, decentralized ledger via sharding.” In: *2018 IEEE symposium on security and privacy (SP)*. IEEE. 2018, pp. 583–598.

[110] Dániel Kondor et al. “Do the rich get richer? An empirical analysis of the Bitcoin transaction network.” In: *PloS one* 9.2 (2014), e86197.

[111] Yana Kortsarts and Jeffrey Rufinus. “Randomized algorithms.” In: *Journal of Computing Sciences in Colleges* 20.5 (2005), pp. 66–67.

[112] Hugo Krawczyk and Tal Rabin. “Chameleon hashing and signatures.” In: (1998).

[113] Heba A Kurdi. “HonestPeer: An enhanced EigenTrust algorithm for reputation management in P2P systems.” In: *Journal of King Saud University-Computer and Information Sciences* 27.3 (2015), pp. 315–322.

[114] Jae Kwon. “Tendermint: Consensus without mining.” In: *Draft v. 0.6, fall* 1.11 (2014).

[115] Frode van der Laak. “Utilise 5G Mobile Handset as DAO and Node in Layer 1 Proof of Authority Blockchain.” In: *Global journal of computer science and technology* (2024).

[116] Leslie Lamport. “Time, clocks, and the ordering of events in a distributed system.” In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 179–196.

[117] Leslie Lamport and Nancy Lynch. “Distributed computing: Models and methods.” In: *Formal models and semantics*. Elsevier, 1990, pp. 1157–1199.

[118] Leslie Lamport, Robert Shostak, and Marshall Pease. “The Byzantine generals problem.” In: *Concurrency: the works of leslie lamport*. 2019, pp. 203–226.

[119] Daniel Larimer. “Delegated proof-of-stake (dpos).” In: *Bitshare whitepaper* 81 (2014), p. 85.

[120] Enrique Larraia, Mehmet Sabir Kiraz, and Owen Vaughan. “How to Redact the Bitcoin Backbone Protocol.” In: *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2024, pp. 485–493.

[121] Bahareh Lashkari and Petr Musilek. “A comprehensive review of blockchain consensus mechanisms.” In: *IEEE access* 9 (2021), pp. 43620–43652.

[122] Jianhao Li et al. “Wolverine: A scalable and transaction-consistent redactable permissionless blockchain.” In: *IEEE Transactions on Information Forensics and Security* 18 (2023), pp. 1653–1666.

[123] Jun Li et al. “Mobile payment with alipay: An application of extended technology acceptance model.” In: *Ieee Access* 7 (2019), pp. 50380–50387.

[124] Xin-Yu Li et al. “Escaping from consensus: Instantly redactable blockchain protocols in permissionless setting.” In: *IEEE Transactions on Dependable and Secure Computing* (2022).

[125] Yeru Liang et al. “A review of rechargeable batteries for portable electronic devices.” In: *InfoMat* 1.1 (2019), pp. 6–32.

[126] *Litecoin - Open source P2P digital currency — litecoin.org*. https://litecoin.org/. [Accessed 18-03-2025].

[127] Bin Luo and Changlin Yang. "AeRChain: An Anonymous and Efficient Redactable Blockchain Scheme Based on Proof-of-Work." In: *Entropy* 25.2 (2023), p. 270.

[128] Jinhua Ma et al. "Redactable blockchain in decentralized setting." In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 1227–1242.

[129] Dahlia Malkhi, Yishay Mansour, and Michael K Reiter. "On diffusing updates in a Byzantine environment." In: *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*. IEEE. 1999, pp. 134–143.

[130] Dahlia Malkhi, Michael Merritt, and Ohad Rodeh. "Secure reliable multicast protocols in a WAN." In: *Proceedings of 17th International Conference on Distributed Computing Systems*. IEEE. 1997, pp. 87–94.

[131] Alexander Marsalek and Thomas Zefferer. "A correctable public blockchain." In: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE. 2019.

[132] Roman Matzutt et al. "A moderation framework for the swift and transparent removal of illicit blockchain content." In: *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2022, pp. 1–9.

[133] Roman Matzutt et al. "A quantitative analysis of the impact of arbitrary blockchain content on bitcoin." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2018, pp. 420–438.

[134] Roman Matzutt et al. "How to securely prune bitcoin's blockchain." In: *2020 IFIP Networking Conference (Networking)*. IEEE. 2020, pp. 298–306.

[135] Roman Matzutt et al. "Thwarting unwanted blockchain content insertion." In: *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE. 2018, pp. 364–370.

[136] Muhammad Izhar Mehar et al. "Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack." In: *Journal of Cases on Information Technology (JCIT)* 21.1 (2019), pp. 19–32.

[137] Xianning Meng et al. "A Conflict-Free Redactable Blockchain." In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE. 2022, pp. 285–290.

[138] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford university, 1979.

[139] Andrew Miller et al. "The honey badger of BFT protocols." In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 31–42.

[140] Du Mingxiao et al. "A review on consensus algorithm of blockchain." In: *2017 IEEE international conference on systems, man, and cybernetics (SMC)*. 2017.

[141] Alexander Mühle et al. "A survey on essential components of a self-sovereign identity." In: *Computer Science Review* 30 (2018), pp. 80–86.

[142] Austin Murphy. "An analysis of the financial crisis of 2008: causes and solutions." In: *An Analysis of the Financial Crisis of* (2008).

[143] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." In: (2008).

[144] Arvind Narayanan et al. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[145] Adam J Oliner et al. "Carat: Collaborative energy diagnosis for mobile devices." In: *Proceedings of the 11th ACM conference on embedded networked sensor systems*. 2013, pp. 1–14.

[146] Emanuel Palm, Olov Schelén, and Ulf Bodin. "Selective blockchain transaction pruning and state derivability." In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE. 2018, pp. 31–40.

[147] Gaurav Panwar, Roopa Vishwanathan, and Satyajayant Misra. "ReTRACe: Revocable and traceable blockchain rewrites using attribute-based cryptosystems." In: *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. 2021, pp. 103–114.

[148] Sunoo Park et al. "Spacemint: A cryptocurrency based on proofs of space." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2018, pp. 480–499.

[149] *Personal Information Protection Law of the People – Republic of China*. http://en.npc.gov.cn.cdurl.cn/2021-12/29/c_694559.htm. [Accessed 22-Jun-2023].

[150] Eugenia Politou et al. "Blockchain mutability: Challenges and proposed solutions." In: *IEEE Transactions on Emerging Topics in Computing* 9.4 (2019), pp. 1972–1986.

[151] Joseph Poon and Thaddeus Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. 2016.

[152] Serguei Popov. "The tangle." In: *White paper* 1.3 (2018), p. 30.

[153] Serguei Popov and Q Lu. "IOTA: feeless and free." In: *IEEE Blockchain Technical Briefs* 6 (2019), p. 964.

[154] Ravi Prasad et al. "Bandwidth estimation: metrics, measurement techniques, and tools." In: *IEEE network* 17.6 (2003), pp. 27–35.

[155] Ivan Puddu, Alexandra Dmitrienko, and Srdjan Capkun. "μchain: How to Forget without Hard Forks." In: *Cryptology ePrint Archive* (2017).

[156] Samyam Rajbhandari et al. "Zero: Memory optimizations toward training trillion parameter models." In: *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2020, pp. 1–16.

[157] Theodore S Rappaport. *Wireless communications: principles and practice*. Cambridge University Press, 2024.

[158] Shafiq Ul Rehman. "Addressing Data Privacy Concerns in Digital." In: *Challenges and Solutions for Cybersecurity and Adversarial Machine Learning* (2025), p. 449.

[159] Team Rocket. *Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies*. 2018.

[160] Phillip Rogaway and Thomas Shrimpton. "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance." In: *International workshop on fast software encryption*. Springer. 2004, pp. 371–388.

[161] Dorit Ron and Adi Shamir. "Quantitative analysis of the full bitcoin transaction graph." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, pp. 6–24.

[162] Michael Rosenberg et al. "zk-creds: Flexible anonymous credentials from zksnarks and existing identity infrastructure." In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2023, pp. 790–808.

[163] Eli Ben Sasson et al. "Zerocash: Decentralized anonymous payments from bitcoin." In: *2014 IEEE symposium on security and privacy*. IEEE. 2014, pp. 459–474.

[164] Sambhav Satija et al. "Blockene: A high-throughput blockchain over mobile devices." In: *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 2020, pp. 567–582.

[165] Johannes Sedlmeir et al. "The energy consumption of blockchain technology: Beyond myth." In: *Business & Information Systems Engineering* 62.6 (2020), pp. 599–608.

[166] Cosimo Sguanci, Roberto Spatafora, and Andrea Mario Vergani. "Layer 2 blockchain scaling: A survey." In: *arXiv preprint arXiv:2107.10881* (2021).

[167] Adi Shamir. "How to share a secret." In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[168] Jun Shen et al. "Verifiable and redactable blockchains with fully editing operations." In: *IEEE Transactions on Information Forensics and Security* 18 (2023), pp. 3787–3802.

[169] Mohammad Shoeybi et al. "Megatron-lm: Training multi-billion parameter language models using model parallelism." In: *arXiv preprint arXiv:1909.08053* (2019).

[170] Victor Shoup. "Practical threshold signatures." In: *International conference on the theory and applications of cryptographic techniques*. Springer. 2000, pp. 207–220.

[171] Yonatan Sompolinsky and Aviv Zohar. "Accelerating bitcoin's transaction processing. fast money grows on trees, not chains." In: *Cryptology ePrint Archive* (2013).

[172] Yonatan Sompolinsky and Aviv Zohar. "Secure high-rate transaction processing in bitcoin." In: *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*. Springer. 2015, pp. 507–527.

[173] Nick Szabo. "Formalizing and securing relationships on public networks." In: *First monday* (1997).

[174] Nick Szabo. "Secure property titles with owner authority." In: *Online at http://szabo. best. vwh. net/securetitle. html* 1 (1998).

[175] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN: 0132392275.

[176] Sabu M Thampi. "Introduction to distributed systems." In: *arXiv preprint arXiv:0911.4395* (2009).

[177] Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. "Blockchain scaling using rollups: A comprehensive survey." In: *IEEE Access* 10 (2022), pp. 93039–93054.

[178] Sri Aravinda Krishnan Thyagarajan et al. "Reparo: Publicly verifiable layer to repair blockchains." In: *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II*. Springer. 2021.

[179] Yangguang Tian et al. "Accountable fine-grained blockchain rewriting in the permissionless setting." In: *IEEE Transactions on Information Forensics and Security* (2023).

[180] Yangguang Tian et al. "Policy-based chameleon hash for blockchain rewriting with black-box accountability." In: *Annual Computer Security Applications Conference*. 2020, pp. 813–828.

[181] Nicolas Van Saberhagen. "CryptoNote v 2.0." In: (2013).

[182] Blesson Varghese et al. "Feasibility of fog computing." In: *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things*. Springer, 2020, pp. 127–146.

[183] Wei Wang et al. "Redactable blockchain based on decentralized trapdoor verifiable delay functions." In: *IEEE Transactions on Information Forensics and Security* 19 (2024), pp. 7492–7507.

[184] Wei Wang et al. "Redactable Blockchain Supporting Rewriting Authorization Without Trapdoor Exposure." In: *IEEE Transactions on Dependable and Secure Computing* (2025).

[185] Wei Wang et al. "Resilient and redactable blockchain with two-level rewriting and version detection." In: *IEEE Transactions on Information Forensics and Security* 20 (2024), pp. 1163–1175.

[186] Wei Wang et al. "Strongly synchronized redactable blockchain based on verifiable delay functions." In: *IEEE Internet of Things Journal* 10.19 (2023), pp. 16778–16792.

[187] Jeffrey Warshaw et al. "Can an algorithm know the" real you"? Understanding people's reactions to hyper-personal analytics systems." In: *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 2015, pp. 797–806.

[188] Moritz Wendl, My Hanh Doan, and Remmer Sassen. "The environmental impact of cryptocurrencies using proof of work and proof of stake consensus algorithms: A systematic review." In: *Journal of Environmental Management* 326 (2023), p. 116530.

[189] Sam Werner et al. "Sok: Decentralized finance (defi)." In: *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*. 2022, pp. 30–46.

[190] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger." In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[191] Chunhui Wu, Lishan Ke, and Yusong Du. "Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain." In: *Information Sciences* 548 (2021), pp. 438–449.

[192] Jing Xu et al. "Redactable proof-of-stake blockchain with fast confirmation." In: *Cryptology ePrint Archive* (2019).

[193] Shengmin Xu et al. "Accountable and Fine-Grained Controllable Rewriting in Blockchains." In: *IEEE Transactions on Information Forensics and Security* 18 (2022), pp. 101–116.

[194] Xiwei Xu et al. "The blockchain as a software connector." In: *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE. 2016, pp. 182–191.

[195] Tao Ye et al. "A Survey on Redactable Blockchain: Challenges and Opportunities." In: *IEEE Transactions on Network Science and Engineering* (2023).

[196] Maofan Yin et al. "HotStuff: BFT consensus in the lens of blockchain." In: *arXiv preprint arXiv:1803.05069* (2018).

[197] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. "Rapidchain: Scaling blockchain via full sharding." In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 931–948.

[198] Di Zhang et al. "Exploring the redaction mechanisms of mutable blockchains: A comprehensive survey." In: *International Journal of Intelligent Systems* (2021).

[199] Di Zhang et al. "Secure Redactable Blockchain With Dynamic Support." In: *IEEE Transactions on Dependable and Secure Computing* (2023).

[200] Rui Zhang, Rui Xue, and Ling Liu. "Security and privacy on blockchain." In: *ACM Computing Surveys (CSUR)* 52.3 (2019), pp. 1–34.

[201] Guohao Zhou et al. "Fine-grained redactable blockchain using trapdoor-hash." In: *IEEE Internet of Things Journal* 10.24 (2023), pp. 21203–21216.

[202] Qiheng Zhou et al. "Solutions to scalability of blockchain: A survey." In: *Ieee Access* 8 (2020).

[203] Runfang Zhou and Kai Hwang. "Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing." In: *IEEE Transactions on parallel and distributed systems* 18.4 (2007), pp. 460–473.